Mise en Œuvre de Techniques d'Analyse/Synthèse de Texture dans un Schéma de Compression Vidéo

Fabien Racapé

▶ **To cite this version:**

Fabien Racapé. Mise en Œuvre de Techniques d'Analyse/Synthèse de Texture dans un Schéma de Compression Vidéo. Traitement du signal et de l'image [eess.SP]. INSA de Rennes, 2011. Français. NNT : . tel-00680826v2

HAL Id: tel-00680826

https://theses.hal.science/tel-00680826v2

Submitted on 25 Apr 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse

THESE INSA Rennes
sous le sceau de l'Université européenne de Bretagne
pour obtenir le titre de
DOCTEUR DE L'INSA DE RENNES
Spécialité : Traitement du Signal et des Images

présentée par
Fabien Racapé

ECOLE DOCTORALE : *MATISSE*
LABORATOIRE : *IETR*

Mise en Œuvre de Techniques d'Analyse/ Synthèse de Texture dans un Schéma de Compression Vidéo

Thèse soutenue le le 14.11.2011
devant le jury composé de :

Atilla Baskurt
Professeur des Universités à l'INSA de Lyon / Président
Patrick Ndjiki-Nya
Docteur Ingénieur à l'institut Fraunhofer HHI / rapporteur
Philippe Salembier
Prof. des Univ. à l'Université Polytechnique de Catalogne / rapporteur
Luce Morin
Professeur des universités à l'INSA de Rennes / examinateur
Jérôme Viéron
Ingénieur Docteur, ATEME/ examinateur
Dominique Thoreau
Ingénieur Docteur, Technicolor / co-encadrant
Marie Babel
Maître de Conférences à l'INSA de Rennes / co-encadrant
Olivier Déforges
Prof. des Univ. à l'INSA de Rennes / Directeur de thèse

Mise en Œuvre de Techniques d'Analyse/Synthèse de Texture dans un Schéma de Compression Vidéo

Fabien Racapé

En partenariat avec

Table des matières

Remerciements

Je tiens d'abord à remercier l'équipe qui m'a encadré et conseillé durant ces trois années. Elle est partagée entre le Groupe Image de l'Institut d'Electronique et de Télécommunications de Rennes et le laboratoire Video Processing and Perception de Technicolor.

Un grand merci d'abord à Olivier Déforges, mon directeur de thèse, dont les conseils, les idées et le soutien n'ont pas faibli durant 3 ans. Merci aussi à Marie Babel, qui a coencadré ces travaux de thèse. Le lecteur lui est par avance redevable de l'allègement du manuscrit, notamment pour le nombre d'occurrences de « permettre » (il en reste).

Je tiens à remercier les trois *chefs* qui se sont succédé pour mon encadrement à Technicolor :

Dominique Thoreau m'a guidé dès le stage de fin d'étude et jusqu'à l'écriture du manuscrit de thèse qu'il a lu et relu. Il m'a aussi littéralement supporté d'interminables minutes pendant mes innombrables défaillances dans les dévers du gymnase Félix Masson. Jérôme Viéron a généreusement accepté l'épreuve d'encadrer mes travaux de thèse après le stage. Inconditionnel de Renaud, il a finalement laissé béton Technicolor, repassant le flambeau à Dominique pour la dernière année. Edouard François m'a accueilli dans son « work package » et a gardé un œil expert sur mes travaux jusqu'à son départ pour Canon. Un petit bémol pour les joggings dans lesquels je souffrais en faisant le double de foulées, la morphologie est injuste.

Cette équipe a constitué un excellent cadre pour mon entrée dans la vie active, tant pour les connaissances apportées que pour les aspects humains et les luttes partagées.

Je tiens à remercier chaleureusement mes rapporteurs MM. Ndjiki-Nya et Salembier ainsi que M. Baskurt, président du jury, qui ont bien voulu porter leur attention sur mes travaux.

Je remercie Jérémy Aghaei Mazaheri et Simon Lefort qui ont contribué avec la manière à ces travaux durant leur période de stage.

Je remercie les membres et anciens membres du laboratoire *Video Processing and Perception*, dirigé par P. Guillotel, pour l'ambiance, la qualité des débats lors des pauses. Merci aux kiteux, nageux, grimpeux, voileux, skieux et footeux pour le sport nécessaire à la compensation d'une journée (semaine ?) de *debug*. Plus précisément, je remercie le gourou Harouna Kabre pour sa maîtrise d'une situation toujours sous contrôle pendant ces

trois années. Il a géré mon évolution du grade de stagiaire de pacotille nouvelle génération à celui d'ingénieur de brousse cosmoplanétaire.

Merci aussi aux membres du groupe image de l'IETR pour leur soutien. Un grand merci particulièrement à Jérôme Gorin et Maxime Pelcat pour les bonnes heures passées dans la salle 225, avec des blind tests à la fin des journées difficiles. Ils reconnaîtront un jour la valeur du film *Le Havre* d'Aki Kaurismäki. Une seconde mention spéciale à Luce Morin qui me fait confiance pour la suite et qui fut une excellente co-conférencière.

Merci aux amis disséminés un peu partout après les études (Paris, Kyoto, Londres, Lyon, La Haie, Zürich, Le Havre...), qui m'ont fait passer des week-ends réfrigérants pour les neurones.

Merci aux parents et la frangine qui sont venus assister à la présentation. Merci bien sûr à toute la famille et la belle famille pour avoir pris soin du petit thésard.

Un énooOOoorme merci enfin à Lucile, ma chérie, qui m'a supporté et chouchouté durant ces trois années.

Introduction générale

Évoquer dans une même phrase synthèse et compression peut paraitre insensé. Si on comprend qu'un sabre laser, dans un film, provient d'une image de synthèse, on voit mal, au premier abord, quelle synthèse peut avoir lieu dans le décodeur sous la télévision, alors que la mi-temps de la finale de la coupe du monde approche. C'est pourtant ce dont il est question dans ce manuscrit. Sur cette télévision dernier cri achetée pour le mondial, la qualité de l'image haute définition est quasi-irréprochable. Cependant, pour qu'elle apparaisse sur l'écran, il a fallu que l'image soit compressée pour être transmise, puis reçue et reconstruite dans le salon. Or cette pelouse, foulée par les 22 acteurs, demande beaucoup de données à transmettre avec un schéma actuel de compression, étant donnés les détails qui y sont contenus. Pourquoi alors ne pas garder uniquement certains échantillons de cette pelouse, si l'œil humain ne fait pas la différence lorsque le reste est synthétisé ? Cette question constitue le point de départ de ces travaux de thèse.

Présentation générale

Les travaux de thèse présentés dans ce document s'inscrivent dans le domaine de la compression des contenus multimédia, images et vidéos. La croissance de la quantité d'informations, demandée par les applications telles que la télévision haute définition ou encore la diffusion de vidéos sur internet, suppose le développement conjoint de deux domaines : la taille physique des canaux de transmission et les outils de compression. Le standard faisant référence actuellement, connu sous le nom de MPEG-4/AVC, permet déjà de réduire le contenu numérique à transmettre, avec un minimum de dégradations visuelles. Il convient notamment au contexte de la télévision numérique haute définition, c'est-à-dire à la transmission et la réception d'images de taille 1920×1080 pixels avec une fréquence de 25Hz. Or les téléviseurs sont maintenant capables d'afficher ces images à des fréquences nettement supérieures. Cependant, la bande passante n'est pas extensible à souhait pour les opérateurs de diffusion. Ainsi ceux-ci émettent le plus souvent en mode entrelacé, qui permet d'afficher deux *trames* contenant la moitié des lignes, à une fréquence de 50Hz sans augmenter la bande passante. Il est donc nécessaire, pour prolonger la progression de la qualité des images, de travailler sur des outils de compression évolués.

En particulier, la technique de codage des textures n'exploite pas pleinement leur contenu. Elle repose sur un découpage en blocs réguliers de l'image, quelque soit le type de zone traitée : contour, texture, mouvement... Les travaux présentés dans ce manuscrit tentent de proposer une alternative permettant de réduire significativement la quantité

d'informations nécessaire à la transmission d'une région texturée. Ce type de schéma vise à sélectionner de petits échantillons de texture, représentatifs de la région plus vaste à laquelle ils appartiennent. Aussi, il ne sera plus nécessaire de transmettre la région dans son intégralité. Le décodeur, à l'aide de ces échantillons, sera capable de reconstruire la zone manquante, en dissimulant au mieux ce processus à l'observateur. De telles approches sont déjà apparues dans la littérature. La figure 1 permet de donner un aperçu du schéma global d'encodage décodage développé. Orientées *contenu* ou *synthèse* suivant les appellations, ces techniques sont en rupture avec les codeurs existants, et sont donc très délicates à mettre au point et à rendre robustes. À l'aide des outils succinctement évoqués dans le paragraphe suivant et détaillés dans les chapitres à venir, ces travaux tentent donc de contribuer à l'élaboration d'un tel schéma de compression.

Contributions

Plusieurs approches de construction d'un schéma global de compression, ainsi que plusieurs outils associés de synthèse et d'analyse de texture, sont présentés dans ce document.

Dans un premier schéma, des échantillons de petite taille, représentatifs du reste des textures, sont détectés afin d'être encodés avec une meilleure qualité que le reste de l'information. Au moment du décodage de la séquence, ces « patchs » servent de source à une étape de raffinement des détails des textures encodées à une qualité moindre. L'intérêt de ce schéma réside dans une redistribution des débits alloués aux régions en fonction de ce qu'elles peuvent propager aux autres au niveau de la reconstruction. Pour cela, l'opération de raffinement potentielle est estimée côté codeur afin de mesurer l'impact de ces patchs et de détecter ceux qui sont capables d'améliorer la qualité globale de l'image reconstruite. Cette méthode a été implantée et testée avec un outil de raffinement directement applicable dans le domaine transformé : les hautes fréquences des patchs appartenant aux zones qui ont été dégradées sont ajoutées par un encodage de moindre qualité.

Malgré des premiers résultats prometteurs, les limites fortes nous ont contraints à revoir la conception du schéma global. Ainsi, il apparaît que tenter de détecter quelques patchs représentatifs, tout en encodant le reste des textures, ne permet pas une réorganisation efficace du débit. Le problème est donc inversé dans la suite : ce sont les régions synthétisables qui ne seront pas transmises, ou du moins sous-résolues afin de préserver du débit. Il faut pour cela localiser et segmenter les régions candidates de manière cohérente avec les outils de codage et de synthèse utilisés. Ce premier schéma non retenu sera détaillé en annexe afin de privilégier la cohérence du manuscrit avec l'approche par synthèse de texture.

Une étape d'analyse est proposée via des outils de segmentation de régions et de caractérisation des textures qui s'y trouvent. La segmentation des images est un vaste domaine qui ne permet pas de définir un outil idéal universel. Notre travail a donc consisté à trouver et utiliser des outils cohérents entre eux et adaptés au contexte de la compression orientée synthèse de texture.

L'utilisation d'outils de synthèse de texture nous a conduits à implanter et tester plusieurs solutions classiques de la littérature [WL00, Ash01, TZL+02, KSE+03, KAK05, HTW07] afin de trouver les plus adaptées à l'approche de compression visée. Ainsi, l'approche de L.Y. Wei et M. Levoy [WL00], fondée sur une construction pixel à pixel, et celle de V. Kwatra [KSE+03], fondée sur l'ajout de patchs, ont été retenues tant pour leurs performances d'un point de vue qualité visuelle que pour la facilité d'adaptation à notre contexte d'étude. La complémentarité de ces deux approches appliquées en fonction du

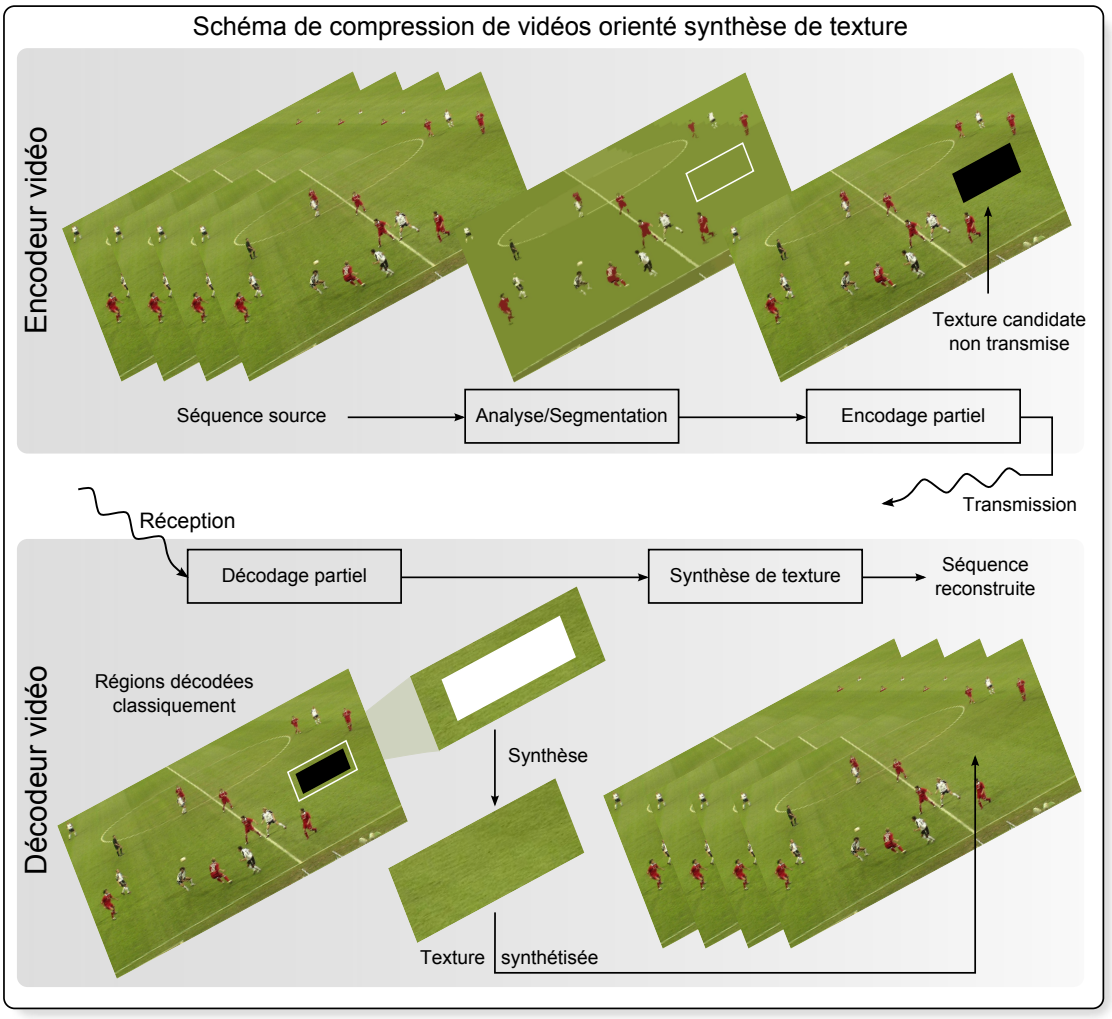


Figure 1 – *Illustration de l'approche proposée*

type de texture traité a motivé l'élaboration d'un schéma permettant d'utiliser l'une des deux méthodes pour chaque région à synthétiser. Dans l'optique d'adapter ce système aux contraintes de la compression, les innovations suivantes ont été proposées :

- la détection automatique des paramètres nécessaires via l'étape d'analyse,
- la construction d'un patch atypique, dédié à la reconstruction de régions contraintes par leur voisinage,
- la définition d'un ordre de synthèse souple et robuste compatible avec les deux approches,
- les décisions sur le type d'algorithme (basé pixel ou patch) en fonction de la texture en cours.

Après la construction, l'optimisation et l'évaluation d'une méthode intra image, un schéma inter image est proposé. L'aspect temporel nécessite alors l'utilisation d'une analyse temporelle à plusieurs niveaux. Un outil d'estimation de mouvement affine permet d'accomplir une segmentation des régions cohérentes en mouvement, mais conduit aussi à une synthèse efficace des régions correctement modélisées. Enfin, une évolution temporelle des outils de synthèse orientés pixel et patch est décrite, permettant de limiter les scintillements dus aux inconsistances temporelles d'une synthèse réalisée image par image.

Organisation du document

Ce document est divisé en deux parties distinctes, elles-mêmes subdivisées en plusieurs chapitres. La première partie présente l'état de l'art des deux grands domaines abordés par ces travaux de thèse, à savoir la synthèse de texture et la compression vidéo. La deuxième partie détaille les contributions des travaux effectués, à savoir l'élaboration de méthodes d'analyse de texture et la construction de schémas de compression orientés synthèse de texture. Le contenu des différents chapitres de ce manuscrit est introduit dans les paragraphes suivants.

Chapitre 1. Une multitude d'algorithmes de synthèse ont vu le jour avec l'essor de l'infographie. Ce chapitre vise à donner un large aperçu de la gamme des algorithmes existants, puis d'en détailler les principales approches. L'efficacité et les résultats en images seront exposés afin d'en évaluer les plus pertinentes pour notre contexte d'étude. Enfin, des techniques d'*inpainting* sont décrites : elles permettent de propager un signal tronqué sur de petites surfaces. Ces méthodes prenant en compte l'environnement de la surface à synthétiser sont importantes dans notre contexte d'utilisation.

Chapitre 2. Nous commencerons par présenter dans ce chapitre les différentes techniques visant à réduire les quantités d'informations à transmettre en exploitant les propriétés du système visuel humain. Les caractéristiques des schémas principaux des standards existants et le futur standard HEVC seront ensuite détaillés. Les métriques permettant de déterminer la qualité de reconstruction des vidéos décodées seront aussi décryptées pour comprendre les problématiques liées aux futurs schémas de compression. Enfin seront présentés les schémas de compression orientés contenu de la littérature. Ce dernier schéma, faisant souvent appel au domaine de la synthèse de texture, constitue l'état de l'art des schémas visés dans ces travaux de thèse.

Chapitre 3. Deux outils principaux se complètent afin d'analyser les séquences source dans l'optique de construire un schéma de compression orienté synthèse : la caractérisation et la segmentation des textures. Les deux premières sections visent à détailler les outils de caractérisation de texture. Après un état de l'art, la construction de descripteurs permettant de déterminer certains paramètres des textures sera proposée. Les deux dernières sections abordent la problématique de la segmentation du contenu. Une méthode complète est proposée, fondée sur des outils de compression, et en lien avec les descripteurs proposés et les algorithmes de synthèse sélectionnés.

Chapitre 4. La construction d'un schéma de compression intra image est ici détaillée. Après la proposition d'une approche de compression orientée raffinement de texture, ses limites orientent les travaux vers un schéma de synthèse de régions segmentées et caractérisées. L'accent est mis sur la cohérence de la chaine ainsi que l'adaptation des différents outils au contexte, notamment des algorithmes de synthèse utilisés.

Chapitre 5. Le développement d'un schéma de compression orienté synthèse fonctionnant sur des séquences vidéo est décrit. Les évolutions nécessaires du schéma retenu pour le chapitre précédent sont détaillées. L'utilisation d'outils d'estimation de mouvement est notamment décrite, afin de segmenter des régions cohérentes temporellement et de compenser en mouvement les régions rigides, dont les variations sont correctement modélisées.

Première partie

Contexte et état de l'art

CHAPITRE 1

Synthèse de textures

Ce chapitre vise à donner un état de l'art des algorithmes de synthèse de texture. Après avoir défini et décrit ce que l'on entend par texture au sens de l'image numérique, les différentes approches seront donc introduites, détaillées et comparées. La multitude des algorithmes de synthèse dans la littérature démontre leur intérêt pour des applications diverses allant de la restauration de contenus altérés [YHS03] aux jeux vidéos en 3 dimensions. Du fait de cette abondance, des choix d'implémentation ont nécessairement été faits. Certaines méthodes seront plus détaillées et analysées du fait qu'elles ont été implémentées dans le cadre de ces travaux. La problématique de cette thèse restant attachée à la compression d'images et de vidéos hors du domaine de la 3D, les extensions d'algorithmes conçues afin de plaquer des motifs sur des objets en 3 dimensions ne seront pas mentionnées.

Après avoir défini les textures dans ce contexte d'étude, les principaux termes utilisés dans le domaine de la synthèse seront introduits. Les principales approches contenues dans la littérature seront enfin détaillées et comparées dans l'optique d'un schéma de compression. Enfin les techniques d'*inpainting*, dédiées au remplissage de blocs d'image à partir du contour, seront développées puisqu'elles s'appuient parfois sur la propagation des textures.

1.1 Préambule : texture et synthèse.

1.1.1 Qu'est-ce qu'une texture ?

Un papier peint, une pelouse, du sable... Les *textures* sont omniprésentes dans notre champ de vision, et le sont par conséquent lorsqu'elles sont capturées par un appareil photo ou une caméra vidéo. Dans ce contexte, on limite d'abord la définition de la texture au sens de la *vision*. La texture a d'abord été définie par l'état d'un matériau qui est tissé (dictionnaire Larousse, 2010). Dans le domaine de l'image numérique, une texture est constituée d'un ensemble de *pixels* définissant une région de l'image possédant certaines caractéristiques statistiques et visuelles. Une région texturée est ainsi définie comme une surface non uniforme qui partage des caractéristiques statistiques qui ne peuvent pas être discernées par la *perception visuelle* [Jul62]. Cette remarque définit la propriété fondamentale d'une texture : la **stationnarité**. Une distribution est par essence stationnaire si sa

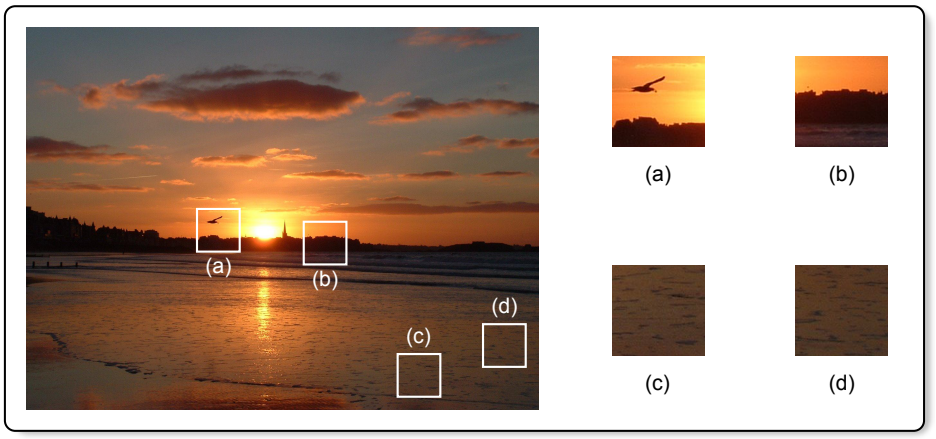


Figure 1.1 – *Régions texturées : la région contenant les fenêtres (c) et (d) est dite texturée, elle répond au critère de stationnarité puisque (c) et (d) apparaissent visuellement similaires. Ce qui n'est pas le cas pour (a) et (b).*

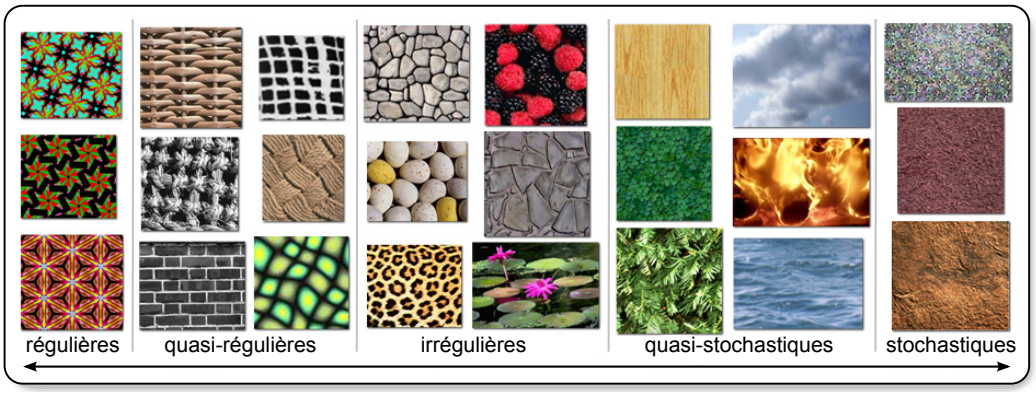


Figure 1.2 – *Gamme des textures suivant leur structure fournie dans [Lin05].*

loi spatiale est invariante par translation. Cependant, dans le cadre de l'image numérique et de la synthèse, on utilisera une définition plus large fondée sur des propriétés visuelles permettant une certaine tolérance sur le critère de stationnarité. Pour illustrer la stationnarité, la figure 1.1 représente une image sur laquelle on pose des fenêtres de même taille en différentes positions. Dans ce contexte, une région sera donc dite stationnaire si l'image parait similaire, observée à travers deux fenêtres situées à différentes positions. Le concept de stationnarité, défini comme tel, introduit une notion de subjectivité, étant fondé sur une propriété visuelle. Il n'est donc pas vu ici comme un critère strictement discriminant.

Ainsi, une large gamme de surfaces peut être perçue comme *texture* qu'il va falloir décrire et caractériser. Décrire une texture consiste à en identifier les caractéristiques visuelles, *i.e.* le contraste, la régularité, la fréquence, la granularité, l'orientation [Law80]... Le paragraphe suivant présente certaines propriétés permettant de classer les textures.

1.1.2 Les différents types de textures.

Certaines surfaces peuvent être définies par l'identification d'un *motif*, *i.e.* une primitive de texture qui, propagée de manière aléatoire ou régulière, permet de construire la surface

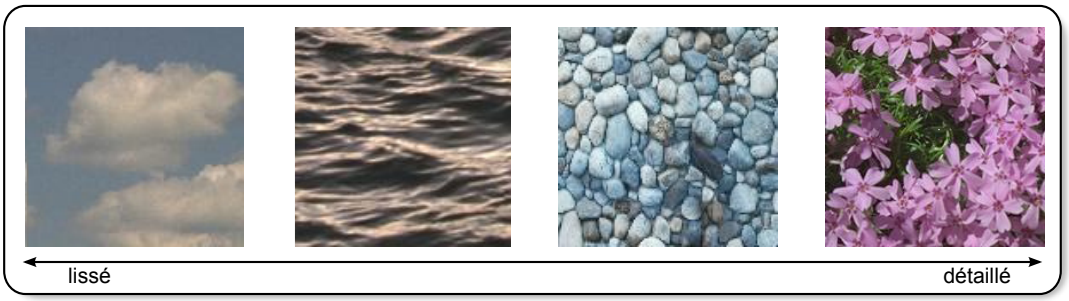


Figure 1.3 – *Gamme des textures suivant le niveau de détails.*

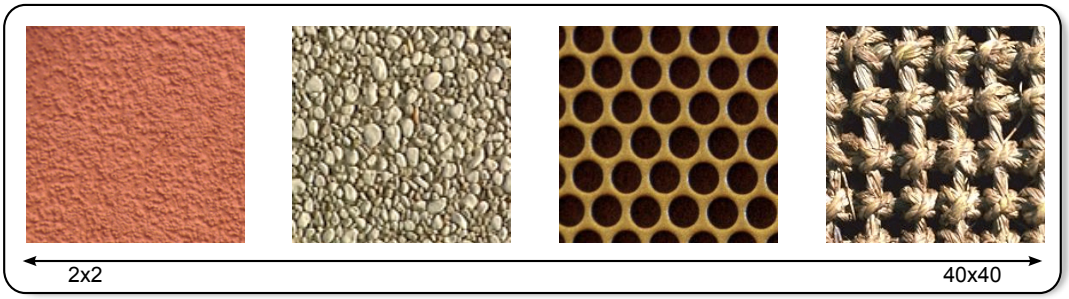


Figure 1.4 – *Spectre des textures suivant la taille des motifs approximée en pixels*

texturée. Néanmoins, il n'est pas possible de déterminer le motif élémentaire d'une surface régie par un modèle purement stochastique. Une classification, décrite dans [LLH04], distingue les textures suivant leur structure, à savoir : texture purement régulière ou texture purement stochastique. La première est définie par la répétition exacte d'un motif élémentaire alors qu'il apparaît impossible, pour la deuxième, de détecter une quelconque primitive. La version illustrée représentée dans la figure 1.2 de cette même classification, apposant les textures de la surface régulière à la surface stochastique, est proposée dans [Lin05]. Cette façon de classer est très pertinente pour la synthèse de texture car elle distingue certains algorithmes plus efficaces pour répéter des motifs de ceux qui vont donner une bonne impression de granularité stochastique stationnaire. Elle permet donc de cibler le synthétiseur en fonction de la texture source, en témoignent les travaux présentés dans [LTL02, LLH04] qui traite uniquement de la synthèse de textures quasi régulières. On peut penser cependant à d'autres classifications :

- La figure 1.3 classe les textures en fonction de leur niveau de *détails*. Ce niveau de détails permet par exemple de distinguer les textures avec de forts gradients et contenant des fréquences spatiales élevées de celles plus lisses, telles que de l'eau ou des nuages seront mieux reproduites par d'autres algorithmes. Cette distinction permet d'adapter le traitement aux caractéristiques du signal.
- Une autre manière de classer peut aussi se fonder sur la taille des motifs à reproduire. Cette taille est bien évidemment essentielle pour le choix ou le paramétrage des algorithmes. La figure 1.4 ordonne les textures suivant la taille du motif élémentaire. Le but est alors de trouver des algorithmes ou des paramétrages permettant de synthétiser la plus large gamme de textures quelle que soit la classification.

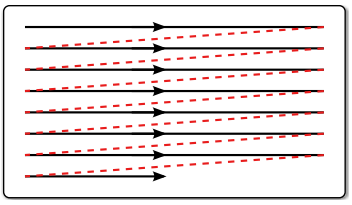


Figure 1.5 – *Ordre de parcours de l'image* raster scan.

1.2 A propos de synthèse.

Un algorithme de synthèse de texture permet de créer une grande surface texturée à partir d'un échantillon source de taille réduite. La généralité de cette définition conduit à une diversité d'applications possibles. Cependant, la plupart des travaux présentés dans ce chapitre se situent principalement dans le domaine de l'infographie. Le domaine des jeux vidéo, par exemple, ou autres simulateurs, sont avides de ce type d'algorithmes : il s'agit en effet d'afficher les différents objets modélisés ou les surfaces texturées, sans avoir à stocker de trop grandes quantités d'information. Certaines applications, comme le *mapping* (ou cartographie) de textures sur des solides en 3 dimensions, ciblent des domaines éloignés de la compression de vidéos. Ces techniques ne seront donc pas abordées dans ce chapitre, focalisé sur la synthèse de texture sur des surfaces 2D.

La synthèse de texture s'inspire directement des modèles probabilistes afin de modéliser la surface de sortie à partir de celle de l'échantillon d'entrée. Plusieurs approches fondamentales de modélisation de la synthèse se confrontent et se divisent en deux grandes catégories.

- L'approche *paramétrique* : elle repose sur la détermination des paramètres du modèle probabiliste. Ces paramètres étant estimés, ils servent directement à l'algorithme de synthèse.
- L'approche *non-paramétrique* : aucun modèle de texture n'est préétabli. La synthèse et la modélisation de la texture de sortie sont conduites de façon simultanée.

Chacune de ces approches possède ses avantages et inconvénients. L'approche paramétrique est efficace : suite à une analyse des paramètres, l'étape de synthèse converge rapidement. Elle est cependant inadaptée à la synthèse de textures qui ne sont pas réellement stationnaires. De légères variations globales peuvent en effet être observées, occasionnant des synthèses défaillantes. L'approche non-paramétrique se révèle, quant à elle, plus laborieuse puisque l'estimation des paramètres est indissociable de l'étape de synthèse elle même. La gamme des textures qu'il est possible de synthétiser est néanmoins plus large.

A partir de ces deux types d'approches, les sections suivantes montrent la difficulté de trier les algorithmes par famille, tant les paramètres à prendre en compte et les stratégies divergent. En effet, certains algorithmes construisent la surface motif à motif, ou pixel à pixel, déterminent directement une loi de probabilité sur la surface entière, ou encore utilisent une approche multirésolution...

A propos du vocabulaire de la synthèse.

- **Patch** : ce terme désigne au départ l'échantillon source de texture pour la synthèse. Cependant, ce terme peut à la fois décrire l'échantillon source de texture mais aussi une sous-partie de celui-ci qui est sélectionnée pour être ajoutée à la surface

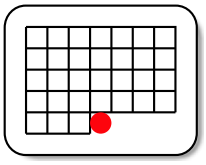


Figure 1.6 – *Forme en L d'un voisinage causal de 7 pixels de côté dans l'ordre* raster scan.

synthétisée. Une confusion pouvant s'installer lors de la synthèse dite *basée patch*, on précisera dans les cas de doute : *patch source* afin d'identifier la texture source.

- **Raster scan** : ordre de parcours d'une image numérique dans l'ordre de la lecture, *i.e.* de gauche à droite avec un retour à la gauche pour le saut à la ligne suivante. Cet ordre, communément utilisé dans le traitement des images numériques, et particulièrement dans le domaine de la compression, est illustré par la figure 1.5
- **Voisinage** : ensemble 2D de pixels contenus dans l'environnement direct d'un pixel ou d'un groupe de pixels considéré. Un voisinage *causal* contient ainsi des pixels déjà reconstruits à l'instant t de traitement du pixel courant. La figure 1.6 illustre un voisinage typique à base carré, dans lequel on n'a gardé que les pixels connus d'après l'ordre de synthèse *raster scan*.

La section suivante introduit les champs Markoviens qui constituent une propriété fondamentale régissant une majorité des algorithmes décrits dans les parties suivantes.

1.3 Les champs de Markov

Les champs de Markov 2D, introduits dans [Bes74], représentent une extension des chaînes de Markov au domaine de l'image ou à un graphe 2D. Dans le contexte d'une image numérique, le champ est constitué d'un réseau S discret et fini de pixels à valeurs dans $\mathbb{N}^2$, pour une image en niveaux de gris. Pour décrire les interactions locales, on définit un système de voisinage $\nu = \{\nu_s \| s \in S\}$ par les deux conditions :

$$\begin{cases} s \notin \nu_s \\ \forall (s,t) \in S \times S \quad s \in \nu_t \Leftrightarrow t \in \nu_s \end{cases} \tag{1.1}$$

Si x est la valeur du pixel à la position s, on définit un champ de Markov X par le fait que la probabilité conditionnelle locale en s n'est dépendante que de son voisinage ν_s. Formellement, on définit $x_s = \{x_t | t \in \nu_s\}$. Les probabilités conjointes associées au champ de Markov X s'écrivent

$$P(X(s) = x_s | X(S \backslash s)) = P(X(s) = x_s | X(\nu_s)). \tag{1.2}$$

Un champ de Markov est donc la réalisation d'un processus stochastique, stationnaire et local. La valeur d'un pixel n'est dépendante que de celles de ses voisins locaux, cette propriété étant indépendante de la localisation du pixel sur la surface synthétisée.

Les champs de Markov se sont révélés efficaces pour une multitude d'approches dans le domaine de l'infographie [KSS80, CJ83, GG86, Li95, DV05], et tout particulièrement pour la synthèse de texture, en témoigne le nombre d'algorithmes présentés ci-après qui s'en inspirent. Par exemple, certains algorithmes utilisent les champs de Markov. Du faitde la propriété suivante : l'ajout de pixels ou de groupes de pixels se fait uniquement à partir de l'étude de leur voisinage précédemment reconstruit. La volonté première de ce type

de solution reste de construire une texture stationnaire visuellement similaire au patch source.

Les premiers algorithmes de synthèse, présentés dans la partie suivante, s'inspirent ainsi directement de processus stochastiques tels que les champs Markoviens. Le but est alors de faire converger certaines propriétés (espérance, moments d'ordres supérieurs...) de la surface synthétisée vers celles du patch source considéré.

1.4 Les approches statistiques

1.4.1 Premières approches de synthèse

Les tous premiers algorithmes de synthèse de texture sont régis par des approches procédurales, c'est à dire par des processus qui transforment un signal prédéfini en texture. Par exemple, les fonctions non-linéaires de Perlin [Per85] permettent de converger vers une texture donnée. R. Chellapa propose lui dans [CK85] l'application d'un modèle autorégressif 2D pour la synthèse de texture. Dans un modèle autorégressif, la valeur d'un pixel synthétisé dépend d'une combinaison de celles de ses voisins dans toutes les directions, additionné d'un bruit blanc additionné, soit

$$I_s = \sum_{r \in N_s} \theta_r I_r + e_r \tag{1.3}$$

avec I_s la valeur du pixel à la position s, N_s le voisinage de la position s, θ_r les paramètres du modèle, et e_r le bruit blanc uniformément distribué. Cette approche paramétrique nécessite donc d'estimer les paramètres θ_r afin de modéliser la texture.

La limitation principale de ces synthétiseurs réside dans le fait qu'il faut construire un synthétiseur par texture. C'est pourquoi de nouvelles fonctions de base ont été ensuite proposées dans [Wor96] afin de créer et d'enrichir les fonctions de base de [Per85] encore appelées *bruit de Perlin*. De plus, B. Julesz a fait l'hypothèse dans [Jul81] que les processus d'ordre inférieur à trois n'étaient pas capables de reproduire une texture naturelle visuellement satisfaisante. L'utilisation des champs de Markov décrits dans la partie précédente est en mesure de répondre à ce critère. C'est pourquoi la plupart des algorithmes suivants s'en sont inspirés.

Un algorithme décrit dans [CCB85] propose ainsi d'appliquer d'abord le modèle des champs markoviens au codage des textures, en utilisant la méthode des moindres carrés afin d'estimer les paramètres du modèle.

Une approche de synthèse combinant champs de Markov et filtres multicanaux est décrite dans [ZWM98]. La synthèse est produite à partir d'un filtrage paramétré d'une surface par un échantillon de départ, appelé FRAME pour Filters, Random Fields and Maximum Entropy. L'effort à produire reste cependant prohibitif en termes de coûts de calcul d'abord par les larges filtres à adopter pour les caractéristiques basses résolutions des motifs. De plus, la convergence de l'échantillonnage n'est pas maitrisée au niveau du nombre d'itérations à appliquer.

Certaines applications utilisent directement des modèles statistiques afin de créer des textures évoluant dans le temps et ainsi créer des séquences vidéo de texture. On peut citer par exemple les travaux présentés dans [LLAY06], qui s'appuient sur une approche non paramétrique. Cette synthèse dynamique de textures est focalisée sur une approche temporelle de l'évolution des paramètres fréquentiels de la texture. D'autres approches

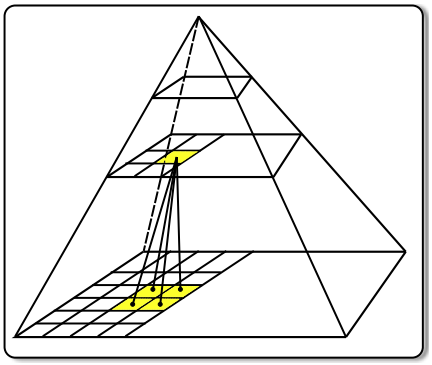


Figure 1.7 – *Pyramide d'images.*

paramétriques [SP95, SDW01, WZ02] permettent aussi de modéliser des variations temporelles. Elles sont malheureusement quasi inutilisables dans le domaine de la reconstruction de régions pour la compression puisqu'elles ne permettent pas de propager les motifs déjà présents autour des régions.

Les algorithmes fondés sur les champs de Markov considèrent donc la probabilité de la valeur d'un pixel uniquement en prenant en compte les pixels voisins. La conception de ce voisinage peut néanmoins poser problème. En effet, dans le cas des textures composées de larges primitives ou de structures atypiques, la considération d'un voisinage très restreint ne suffit pas. D'un autre côté, l'utilisation d'approches non-paramétriques, capables de synthétiser une large gamme de textures puisque le paramétrage est lié au processus de synthèse, s'avère fastidieuse lors de la considération de larges voisinages. C'est pourquoi les algorithmes de la partie suivante se proposent de pallier ces défaillances en utilisant des approches pyramidales.

1.4.2 Approches multirésolution et ondelettes

L'approche multirésolution ou pyramidale repose sur le fait de construire une pyramide, c'est à dire un édifice constitué à la base de l'image source définissant le niveau 0, et de $N-1$ versions sous-résolues, N étant le nombre d'étages de la pyramide. La structure classiquement utilisée est dite *dyadique*, puisque les dimensions de chaque étage d'indice n suivent

$$\begin{cases} Nb_{Col}(n) = \frac{Nb_{Col}(0)}{2^n} \\ Nb_{Lig}(n) = \frac{Nb_{Lig}(0)}{2^n} \end{cases}, \tag{1.4}$$

où Nb_{Col} et Nb_{Lig} correspondent respectivement au nombre de colonnes et de lignes. Par construction, chaque niveau n de la pyramide est obtenu en conservant un pixel sur 4 de l'image au niveau $n-1$, comme illustré sur la figure 1.7. La différence entre les types de pyramides réside dans le filtre qui permet de passer d'un étage donné à l'étage supérieur, comme par exemple le filtrage gaussien pour les pyramides du même nom. Cette structure a l'intérêt de permettre la manipulation séparée des basses fréquences spatiales et des détails. Ainsi, il sera possible de commencer à synthétiser les larges structures grâce aux étage supérieurs de la pyramide, puis d'affiner les détails par le traitement de l'étage 0 définissant l'image de sortie en pleine résolution.

La méthode de R. Paget et I. D. Longstaff décrite dans [PL95] se propose elle d'utiliser un modèle non paramétrique fondé sur les champs de Markov. Cette méthode est inspirée

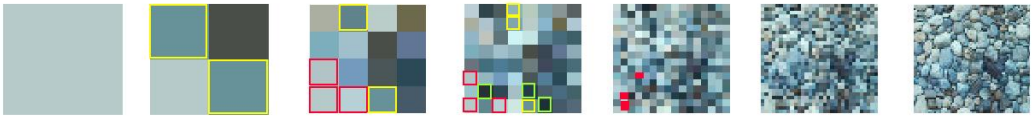


Figure 1.8 – *Approche hierarchique de synthèse de texture [DB97].*

des modes conditionnés itératifs ou ICM décrits dans [Bes86] qui permettent une relaxation déterministe du champ, sensée trouver en chaque position un maximum local de la fonction de densité de probabilités (ou PDF) jointe. Cette méthode utilise une approche multirésolution où la pyramide est synthétisée de la résolution la plus basse jusqu'à la pleine résolution. A chaque fois qu'un équilibre est trouvé en utilisant les ICM à un niveau donné, l'image est sur-échantillonnée et prise comme initialisation pour le niveau suivant. Cette méthode donne des résultats visuels perfectibles mais intéressants sur les textures de la base de Brodatz [Bro99].

Un nouveau schéma multirésolution est présenté dans [HB95], fondé sur le raffinement itératif des histogrammes des sous-bandes d'une pyramide construite à partir de filtrages linéaires. La surface à synthétiser est initialisée par un bruit blanc. Pour chaque itération, les pyramides de l'échantillon source et de sortie sont construites. L'histogramme de chaque sous-bande de sortie est ensuite apparié à la sous-bande correspondante de la pyramide calculée à partir de la source. Cette technique produit de bons résultats pour la synthèse de textures stochastiques. Elle reste cependant perfectible, comme les méthodes précédentes, pour les motifs plus structurés.

Utilisation des ondelettes

Les ondelettes qui ont inondé le domaine du traitement d'image par leurs performances dans de nombreux domaines, n'ont pas fait l'impasse sur la synthèse de texture. Une approche multirésolution de synthèse utilisant une structure hiérarchique est développée dans [DB97]. Ce schéma s'appuie sur le fait que les images texturées, lorsqu'elles sont sous-échantillonnées, contiennent des régions peu différentes les unes des autres. Ainsi, les auteurs soutiennent qu'une réorganisation des basses fréquences, en manipulant une version sous-échantillonnée plusieurs fois, et en laissant inchangées les hautes fréquences, ne va pas changer les caractéristiques visuelles principales de la texture. La figure 1.8 met en évidence des régions qui sont potentiellement interchangeables à certains niveaux de résolution tout en gardant les hautes fréquences, ce qui revient pour cette texture à déplacer des galets entiers.

De la même manière, J. Portilla et E. Simoncelli dans [PS00b], utilisent une transformée en ondelette. Les coefficients sont ensuite modélisés par un histogramme afin d'estimer l'image de sortie, bande par bande. Cette méthode produit des résultats visuels les plus convaincants par rapport aux autres approches paramétriques. On peut encore citer [Men01] qui propose une méthode non paramétrique utilisant la création d'une pyramide fondée sur la transformée en ondelettes discrètes. L'apparition d'un coefficient à un étage de la pyramide dépend directement de l'apparition des coefficients correspondants aux niveaux supérieurs.

L'utilisation de pyramides a permis une avancée considérable dans l'amélioration notamment des approches non-paramétriques, qui peinaient à reproduire de larges motifs de par la localité inhérente aux modèles stochastiques. Elle a d'ailleurs été reprise dans les

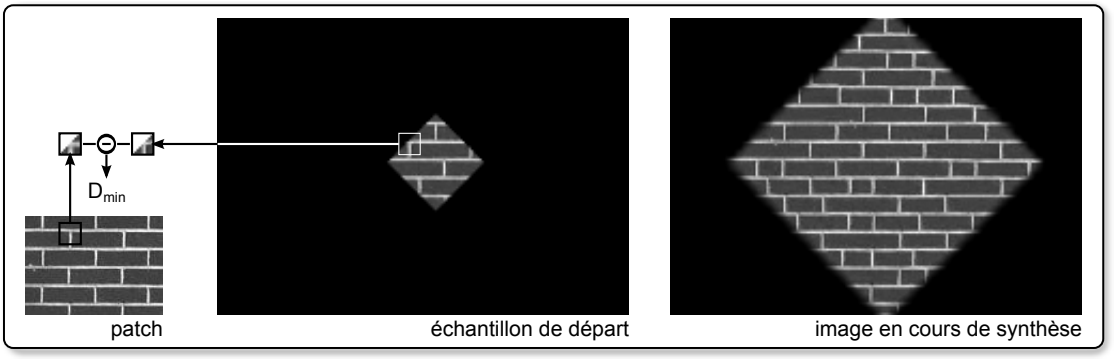


Figure 1.9 – *Méthode de A. Efros pour la recherche du meilleur représentant dans le patch.*

algorithmes présentés dans la suite, qui proposent de construire des surfaces en ajoutant un à un des éléments de la texture source.

1.5 La synthèse basée pixel

Les méthodes décrites dans cette partie sont dites *basées pixel* dans la mesure où la surface de sortie est synthétisée par un processus ajoutant un pixel à la fois, jusqu'au remplissage total de la surface. Il ne s'agit donc plus de regarder directement la surface synthétisée dans son ensemble et de filtrer le signal, mais de construire une texture, pixel à pixel, la manière de choisir les pixels créant la diversité des algorithmes suivants.

1.5.1 Efros et Leung

Le premier schéma de synthèse inspiré des champs de Markov est décrit dans [EL99]. L'idée de base de ce synthétiseur est illustrée par la figure 1.9. La surface synthétisée est initialisée avec un échantillon de 3×3 pixels provenant d'une position prise aléatoirement dans le patch source. Une liste, triée aléatoirement, des pixels adjacents permet ensuite de faire grandir la région initiale jusqu'à atteindre la taille de surface de sortie désirée. La synthèse de chaque pixel de sortie nécessite la définition dynamique de son voisinage, c'est à dire un ensemble de pixels déjà reconstruits appartenant à une fenêtre de taille réglable par l'utilisateur. Ce voisinage est ensuite comparé à tous ceux contenus dans le patch suivant la norme Euclidienne ou L2, *i.e.* la Somme des Carrés des Écarts (SCE) (SSE en anglais). Une liste des voisinages du patch retournant une erreur inférieure à un seuil donné est alors constituée. Un de ces voisinages est finalement aléatoirement choisi et la valeur de son pixel central est alors affectée à la position courante.

Bien qu'élégante par sa simplicité et des résultats visuels encourageant sur certains types de textures, présentés sur la figure 1.10, cette méthode possède quelques limitations. Étant donné l'ordre de la synthèse et la forme changeante des voisinages suivant les pixels disponibles, certaines configurations, comme les textures régulières aux larges motifs, peuvent produire une synthèse médiocre. La technique suivante utilise une forme et une taille de voisinage fixes, afin de pallier ce problème.

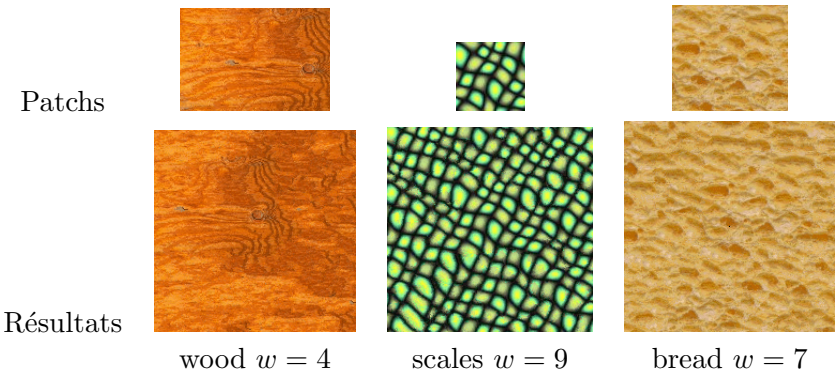


Figure 1.10 – *Résultats de la synthèse de [EL99]. w correspond à la taille en pixels du côté du voisinage carré utilisé.*

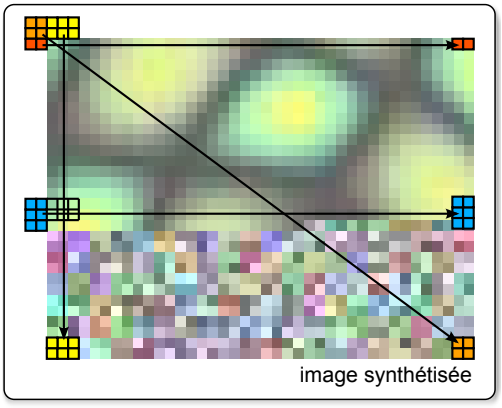


Figure 1.11 – *Périodisation de l'image pour la synthèse des pixels de bords en haut et à gauche.*

1.5.2 L'approche de L.Y. Wei et M. Levoy

Synthétiseur basique

Cette technique, développée dans [WL00], reprend l'idée de synthétiser un à un les pixels via une comparaison de leur voisinage causal *i.e.* les pixels précédemment reconstruits appartenant à un voisinage spatial de taille définie par l'utilisateur. Cependant, plusieurs différences fondamentales permettent de pallier les défaillances du schéma décrit dans [EL99].

- L'image de sortie est synthétisée dans l'ordre *raster scan* (figure 1.5).
- La taille du voisinage de comparaison est fixe durant la synthèse, et par conséquent la forme grâce à l'ordre de synthèse.
- Les pixels de la surface de sortie sont initialisés aléatoirement afin d'amorcer la synthèse. Pour ce faire, l'image de sortie est *périodisée* afin que les premiers voisinages de taille fixe se fondent sur les pixels initialisés aléatoirement. La figure 1.11 illustre la création des premiers voisinages.
- Le pixel choisi pour la position courante est le centre du voisinage minimisant la distance avec celui du pixel courant, après une recherche exhaustive sur le patch. Cette étape est illustrée par la figure 1.12. Le critère de sélection choisi est le même que dans [EL99], à savoir la SCE.

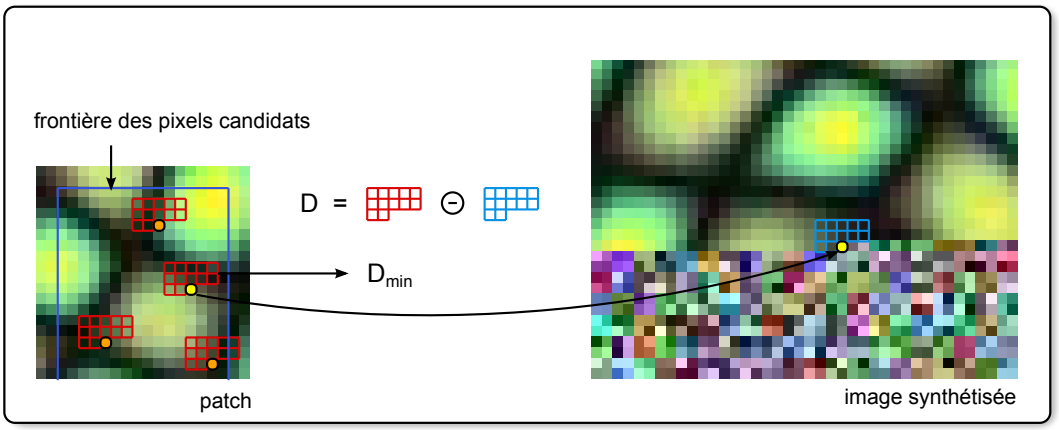


Figure 1.12 – *Méthode de L.Y Wei et M. Levoy : recherche du meilleur représentant dans le patch.*

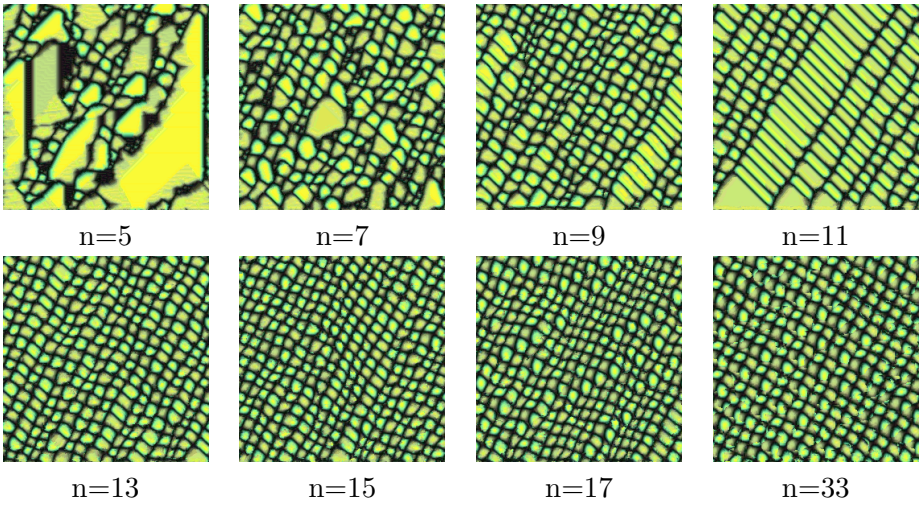


Figure 1.13 – *Evolution des résultats de l'algorithme de L. Y. Wei en fonction de la taille du voisinage.*

La synthèse est amorcée sur quelques lignes avec des voisinages fondés sur un bruit d'initialisation, le nombre de lignes étant égal à la moitié de la taille du voisinage choisie.

La taille fixe de voisinage de comparaison est ici le seul paramètre à définir par l'utilisateur. Empiriquement, il doit être assez grand pour être capable de contenir un motif élémentaire de la texture, mais aussi assez petit par rapport au patch pour permettre un nombre élevé de pixels candidats. La figure 1.13 illustre le rapport crucial entre la taille du voisinage et la qualité de la synthèse appiquée à la mosaïque verte caractéristique : sachant qu'un motif de la texture mesure environ 12×12 pixels, on s'aperçoit que la synthèse est acceptable à partir d'une taille minimale de voisinage de 13×13 pixels. Le résultat de la synthèse avec un voisinage de 33×33 pixels illustre a contrario qu'une trop grande taille de voisinage par rapport à celle du patch (ici 64×64) conduit à une synthèse médiocre.

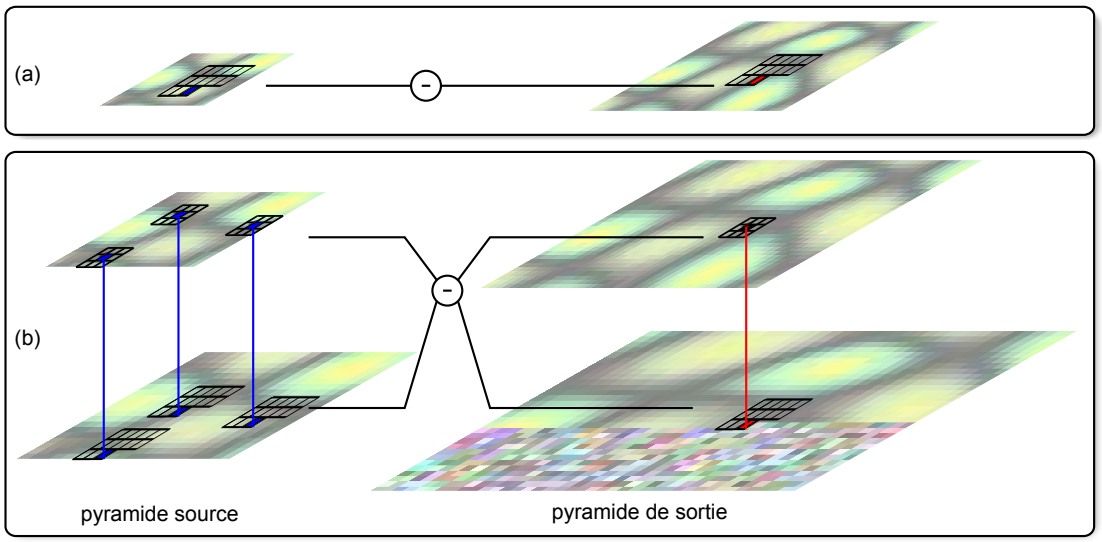


Figure 1.14 – *Algorithme de Wei et Levoy, version multirésolution. (a) synthèse du haut de la pyramide à la manière de l'algorithme basique. (b) la synthèse des étages supérieurs est faite en comparant un voisinage multirésolution : concaténation du voisinage causal du pixel courant et du voisinage carré du pixel à l'étage supérieur dont il hérite.*

Approche pyramidale

Dans le même article [WL00], les auteurs proposent une évolution fondée sur la synthèse d'une pyramide d'images. Le schéma utilise une pyramide Gaussienne. Ainsi, l'initialisation est maintenant constituée d'une étape supplémentaire : la construction d'une pyramide gaussienne pour le patch ainsi que pour l'image de sortie aléatoirement initialisée, le nombre d'étages de la pyramide étant défini par l'utilisateur. La synthèse est ensuite conduite de la résolution la plus grossière à la résolution la plus fine. Le dernier étage de la pyramide est d'abord synthétisé via la méthode de base, en considérant le dernier étage de la pyramide *patch* comme entrée. Les étages suivants sont synthétisés avec un voisinage multirésolution illustré par la figure 1.14, il est constitué de la concaténation du voisinage causal à la résolution courante et d'un voisinage carré proportionnellement *colocalisé* à l'étage supérieur, lui entièrement synthétisé.

La figure 1.15 montre la qualité accrue de la synthèse opérée sur la mosaïque verte avec des tailles de voisinages de 5×5, même en comparant avec la version simple résolution utilisant des voisinages larges. L'algorithme suivant est une optimisation de celui de [WL00], fondé sur l'observation que pour la plupart des textures, la recopie de motifs ou de parties de motifs augmente la qualité de la synthèse.

1.5.3 Schéma de M. Ashikhmin.

Cet algorithme, inspiré des travaux présentés dans [WL00] apporte une accélération ainsi qu'une amélioration des résultats sur certains types de textures. Après avoir décrit le fonctionnement et les résultats de l'algorithme, des résultats en images sont présentés, ainsi qu'une extension du schéma.

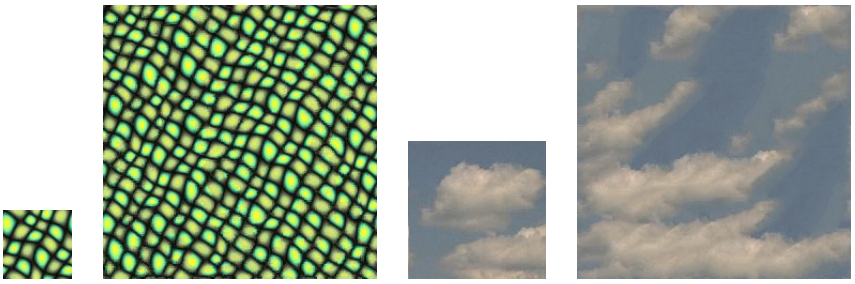


Figure 1.15 – *Résultats de l'algorithme de L. Y. Wei multirésolution utilisant une pyramide de 3 niveaux et un voisinage (7;5).*

Description de l'algorithme

Une extension intéressante pour la reconstruction de régions texturées est proposée dans [Ash01]. Partant de la comparaison de voisinages décrite précédemment, ce nouvel algorithme propose de réduire le nombre de comparaisons. En effet, au lieu de comparer tous les voisinages du patch, seules certaines positions candidates sont testées. Ces positions correspondent au lieu de provenance des voisins du pixel courant comme illustré sur la figure 1.16. Une fois ces positions relevées, la figure 1.16 montre les voisinages correspondants ainsi que les pixels candidats. Cette méthode nécessite donc de stocker la position dans le patch des pixels choisis pour les pixels déjà synthétisés. Un tableau aux dimensions de l'image de sortie et contenant des coordonnées de provenance dans le patch de chaque pixel synthétisé est alors nécessaire. L'algorithme procède alors comme suit :

1. Initialisation aléatoire des coordonnées du tableau et copie, sur l'image de sortie, des valeurs correspondantes dans le patch.
2. Pour chaque pixel de l'image de sortie dans l'ordre *raster scan*, son voisinage est construit et appliqué au tableau de coordonnées.
3. Pour chaque système de coordonnées rencontré dans ce voisinage :
   - le voisinage causal est construit dans le patch centré à la position donnée par les coordonnées,
   - la distance L2 est calculée entre ce voisinage et le voisinage du pixel courant dans l'image de sortie,
   - si la distance est minimisée et si le candidat est éloigné des bords du patch, la valeur du pixel candidat est copiée dans l'image de sortie et ses coordonnées sont stockées,
   - si les candidats sont trop proches des bords du patch, les candidats sont alors réaffectés aléatoirement dans le patch.

Résultats et comparaison avec [WL00]

La figure 1.17 montre que sur les fleurs, les résultats sont meilleurs que [WL00]. Par construction, cet algorithme a l'avantage d'accélérer la recherche dans le patch. De plus, il apporte une amélioration visuelle des résultats sur certaines textures détaillées. Cependant, pour les textures plus lisses, comme la mosaïque verte, l'algorithme présenté dans [WL00] évite la création de frontières visibles. Ces derniers artefacts proviennent de la réaffectation aléatoire une fois qu'une partie du patch a été recopiée et que les candidats possibles sont en bord de patch.

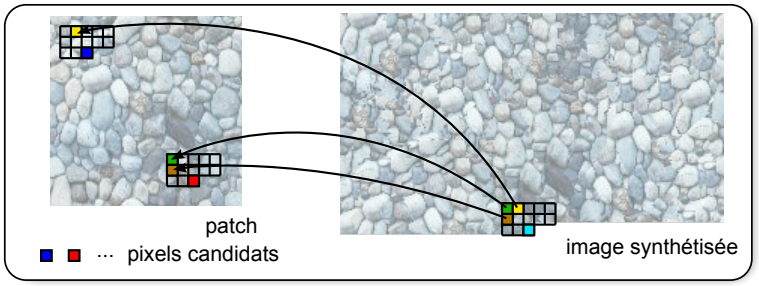


Figure 1.16 – *Méthode d'Ashikhmin pour la recherche du meilleur candidat pour synthétiser le pixel courant.*

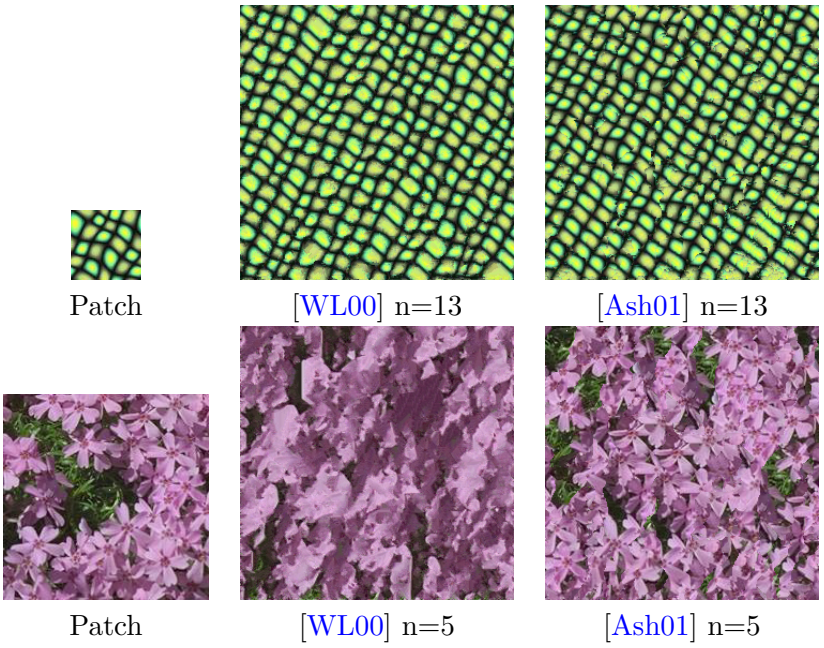


Figure 1.17 – *Comparaison des résultats des algorithmes provenant de [WL00] et [Ash01].*

Synthèse guidée

Ashikhmin propose de remplacer le voisinage causal utilisé par un voisinage carré lors de la synthèse [Ash01]. Ainsi, sur l'image à synthétiser, le voisinage contient toujours une partie causale, contenant des pixels déjà synthétisés, qui permet de prolonger les motifs. Mais il contient aussi des pixels non synthétisés dans la partie non causale, qui vont dépendre de l'initialisation de la surface à synthétiser. Si cette dernière contient un guide de couleur comme illustré sur la figure 1.18, cette partie va influencer la recherche, au moment de calculer la différence L2 avec les voisinages du patch, eux mêmes carrés. Une région verte, comme présenté sur la figure va donc *guider* la recherche vers les zones comportant de l'herbe dans le patch. L'application présentée par la figure 1.19 peut sembler anodine, mais la technique peut s'avérer pertinente dans le cadre des travaux présentés dans ce manuscrit. En effet, le raffinement de textures qui auraient été altérées par le processus de compression, peut être approché par une méthode de ce type, prenant aussi en compte les pixels déjà présents, mais préservant aussi la continuité des motifs grâce à

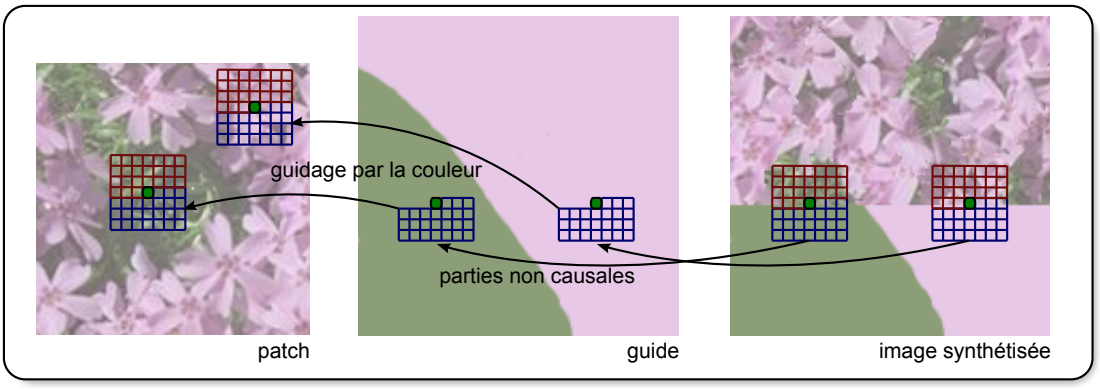


Figure 1.18 – *Méthode d'Ashikhmin pour la recherche du meilleur candidat pour synthétiser le pixel courant.*

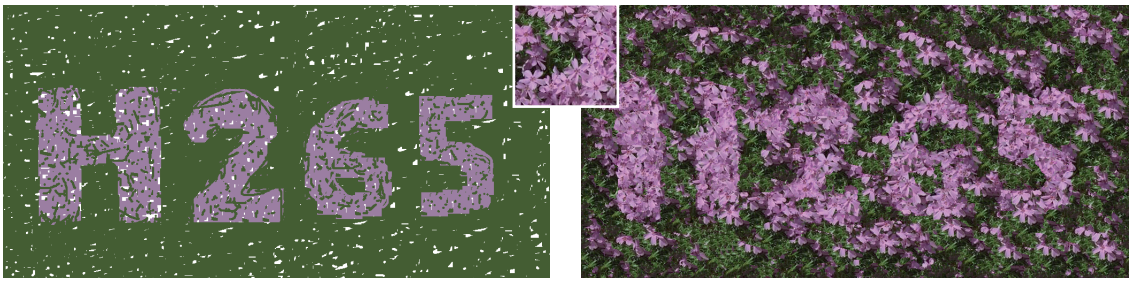


Figure 1.19 – *Application de synthèse guidée proposée par Ashikhmin avec : l'initialisation de l'image de sortie à gauche, le patch au centre et l'image synthétisée à droite.*

la partie causale.

L'algorithme suivant permet de combiner les algorithmes de Wei et Ashikhmin, afin d'obtenir un compromis pour l'utilisateur entre les méthodes, suivant les paramètres à définir.

1.5.4 La k-cohérence proposée par Tong et al.

Description de l'algorithme

Cet algorithme, proposé dans [TZL+02], est inspiré des algorithmes précédents, notamment celui de [Ash01]. Il introduit en plus une étape de prétraitement permettant de connaître les pixels du patch qui se ressemblent. Lors de cette étape, pour chaque pixel du patch, une liste est dressée des coordonnées des $k-1$ pixels du patch qui lui ressemblent le plus au sens de [WL00], c'est à dire en comparant leurs voisinages. Cette recherche exhaustive est néanmoins censée ajouter un faible temps de calcul étant donné la petite taille du patch. Fort de cette étape, l'algorithme procède ensuite à la synthèse dans l'ordre de celle présentée dans [Ash01], en ajoutant dans la recherche des pixels candidats, la liste des pixels *cohérents* pour chaque position retournée par le tableau liant les pixels synthétisés et leurs pixels parents dans le patch. La figure 1.20 montre la nouvelle liste des candidats testés. La figure 1.21 permet ensuite d'observer l'influence du paramètre k et de faire le lien avec les résultats d'Ashikhmin qui correspond à $k = 1$ et Wei qui correspond à un k grand.

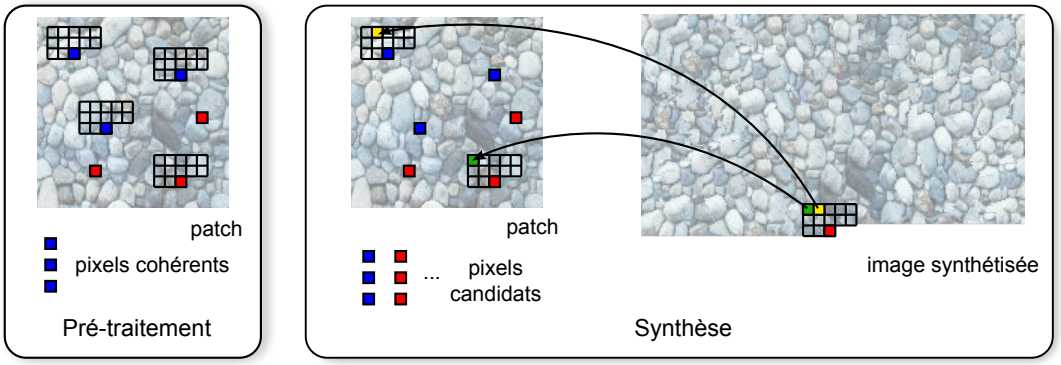


Figure 1.20 – *Schéma de la synthèse par la méthode de k-cohérence.*

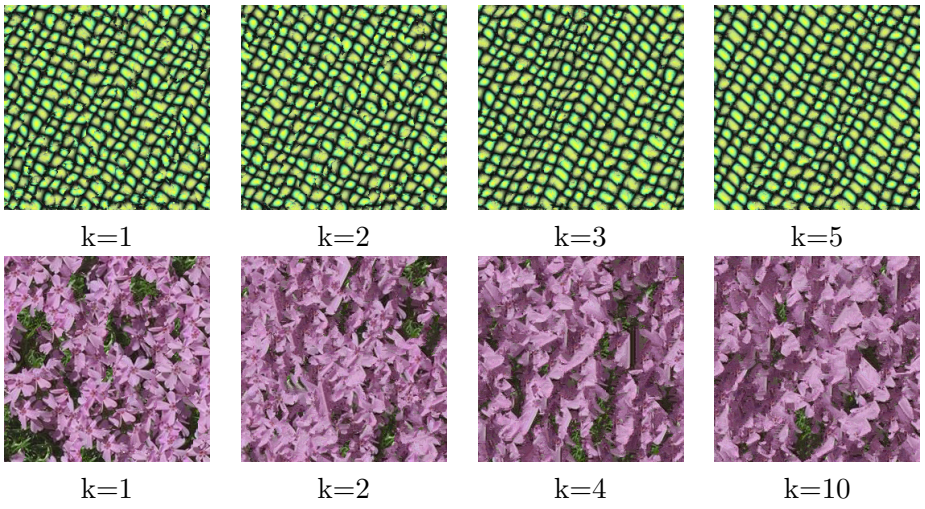


Figure 1.21 – *Résultats de la synthèse par k-cohérence suivant le paramètre k.*

La section suivante liste plusieurs optimisations dans la lignée de ces algorithme *basés pixel.*

1.5.5 D'autres optimisations apportées à ce type d'algorithmes

L'algorithme de L.Y. Wei et M. Levoy présente des résultats qui ne sont pas satisfaisants pour les textures structurées. De plus, les résultats des algorithmes précédents affichent un inéluctable compromis entre qualité et vitesse de synthèse.

L. Y. Wei a proposé un schéma dans [WL02] inspiré de ses précédents travaux présentés dans [WL00] ainsi que ceux de [TZL+02], *i.e.* une synthèse pyramidale et accélérée par la k-cohérence. La nouveauté vient du fait que la synthèse ne dépend pas de l'ordre dans lequel est synthétisée l'image de sortie. La pyramide de sortie est toujours initialisée au niveau de son sommet, sous la version de résolution la plus faible. L'ordre arbitraire permet de synthétiser des pixels aux étages inférieurs avant même la fin de la synthèse complète d'un étage. Les dépendances lors des comparaisons de voisinage multirésolutions ne prendront en compte que les pixels connus. Mis à part l'amélioration des performances de calculs, les résultats en image ne montrent malheureusement pas d'avancée significative en termes de

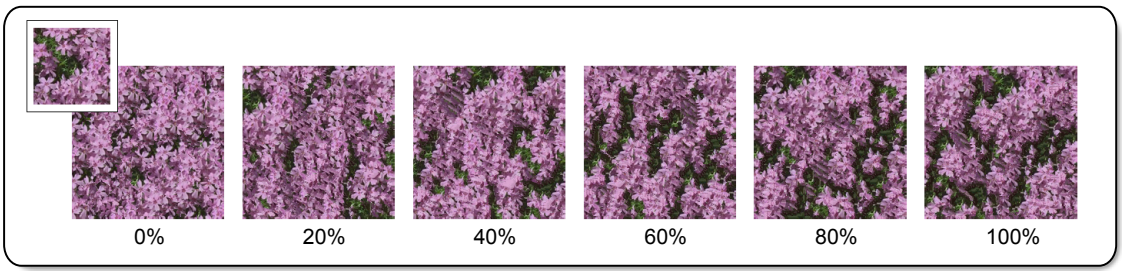


Figure 1.22 – *Résultats de l'algorithme de [SPD07] en fonction de la tolérance appliquée.*

qualité visuelle.

M. Sabha propose dans [SPD07] un algorithme contenant une étape supplémentaire afin d'améliorer la synthèse des textures à motifs structurés, tout en accélérant la recherche des candidats dans le patch. Cette méthode apporte une amélioration sensible de l'algorithme de Wei et Levoy comparable à celle apportée par Ashikhmin dans [Ash01]. Cependant, cet algorithme ajoute une étape simple, et non contraignante, contrairement à celle d'Ashikhmin qui nécessite des conditions aux bords telles que des réaffectations aléatoires lorsque les candidats choisis se trouvent sur les bords du patch. La principale contribution réside dans l'étude préliminaire du voisinage direct du pixel courant, soient les pixels directement adjacents, au nombre de quatre dans l'exemple du voisinage causal présenté en figure 1.6. La condition pour que le candidat considéré dans le patch soit retenu est qu'il possède au moins un pixel de ce voisinage restreint de valeur égale à son correspondant. Une liste des candidats restants est ensuite prise en compte pour une recherche du meilleur au sens de l'algorithme de Wei. De la même manière que le compromis opéré par l'algorithme de k-cohérence entre les algorithme de Wei et Ashikhmin, les auteurs proposent d'introduire un paramètre de tolérance sur la contrainte ajoutée aux pixels du voisinage direct. La figure 1.22 présente l'évolution des résultats en fonction de cette tolérance. Ainsi pour une tolérance de 100%, l'algorithme correspond à la synthèse de Wei et Levoy, efficace pour les textures lissées ; au contraire, une tolérance nulle donne des résultats proches de ceux d'Ashikhmin. En effet, l'idée est semblable puisqu'elle incite l'algorithme à prolonger la texture avec un lien fort dans au moins une des directions.

La partie suivante présente une évolution des algorithmes pyramidaux fondée sur l'étude des approches développées dans [WL02] et [TZL+02].

1.5.6 Une approche pyramidale atypique

Cet algorithme développé par S. Lefebvre et H. Hoppe dans [LH05] est fondé sur l'utilisation d'une pyramide Gaussienne. On parlera plutôt de *pile Gaussienne* puisque l'algorithme intègre un héritage des coordonnées de provenance dans le patch source entre les étages. L'algorithme 1.1 ainsi que la figure 1.24 représentent les différentes étapes du processus. Comme pour l'approche de [WL00], la pile P de sortie est construite de la version la plus grossière jusqu'à la pleine résolution.

La figure 1.23 décrit la manière de sur-échantillonner la version au niveau $n-1$ en prenant les coordonnées du patch correspondant au niveau n. L'étape de *Jitter* correspond à une modification aléatoire des coordonnées des antécédents dans un périmètre donné par l'utilisateur, ici ± 1 dans les deux directions de l'image. Une étape itérative de raffinement

Algorithme 1.1: Algorithme de synthèse par H. Hoppe

Entrées : Patch source S, paramètres
Sorties : Pile de sortie $\{P_l | l = -1, \ldots, L\}$

Construire Pyramide Gaussienne Pyr_S ;
$P_{-1} = (0..0)^T$; // initialisation à zéro des coordonnées
pour $l = 0 : L$ // *pour chaque étage*
faire
  P_l =sur-échantillonnage(P_{l-1}) ;
  P_l =Jitter(P_l) ; // perturbation des coordonnées
  si $l > 2$ **alors**
    pour $i = 1 : c$ // *quelques corrections*
    faire
      P_l =Raffinement(S_l);
    fin
  fin
fin
retourner P_L;

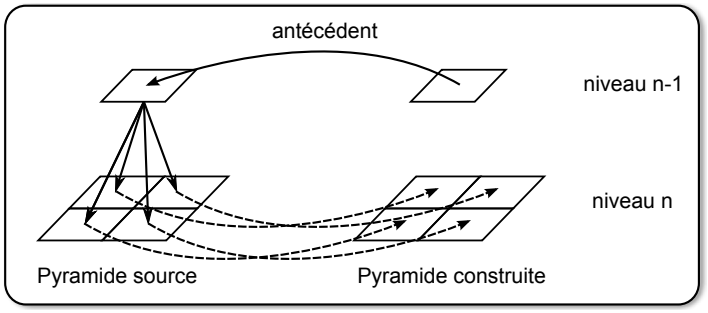


Figure 1.23 – *Méthode de sur-échantillonnage à partir de la pyramide source.*

est ensuite appliquée : elle correspond à une recherche de type k-cohérence permettant de mettre à jour le tableau des antécédents.

1.5.7 Conclusion sur les approches pixels

Les approches pixel constituent donc une avancée essentielle pour les approches non paramétriques, permettant de synthétiser une large gamme de textures avec les mêmes caractéristiques algorithmiques. Certaines restent cependant lourdes au niveau du temps de calcul et des compromis sont donc à faire entre qualité et optimisation.

Ashikhmin [Ash01] et Sabha [SPD07] ont dessiné des algorithmes qui favorisent la recopie de motifs ou de parties de motifs, pixel à pixel, afin de pallier les inconsistances de synthèse des textures structurées. Les algorithmes décrits dans la partie suivante prennent eux le parti de copier directement des zones entières du patch source.

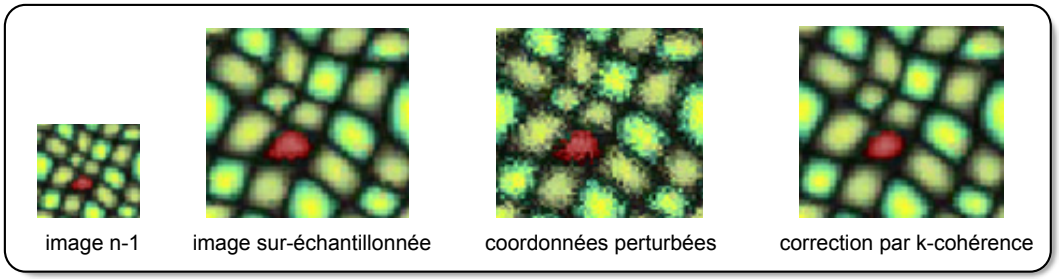


Figure 1.24 – *Illustration de la synthèse pyramidale de [LH05].*

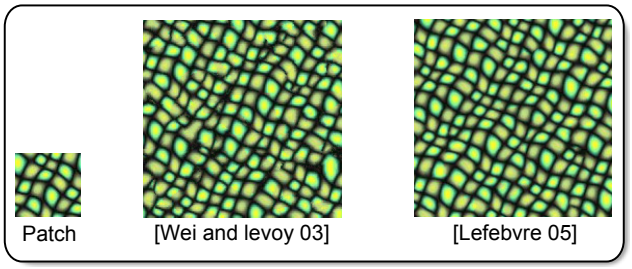


Figure 1.25 – *Comparaison des résultats de [WL00] et [LH05].*

1.6 La synthèse basée patch

La synthèse basée pixel ayant montré des limites au niveau du compromis entre rapidité et qualité, les algorithmes suivants prennent le parti de travailler la synthèse patch par patch. Un *patch* dénote ici un groupe de pixels formant une surface continue. Il peut donc s'agir d'une partie ou d'une sous-partie de l'échantillon source dans son intégralité. Pour éviter toute confusion, on appellera donc *échantillon source* la texture en entrée de l'algorithme de synthèse, afin de la dissocier du patch, qui peut en être une sous-partie ou la totalité. Le fait de copier des régions entières amène un nouveau défi par rapport à la copie pixel à pixel : la gestion des contraintes entre les régions ainsi copiées. Ce défi va engendrer la diversité des approches proposées ci-après.

1.6.1 Approches aléatoires

Un des premiers travaux dans ce sens est proposé dans [PFH00]. Ce schéma s'adresse à la synthèse de texture sur des volumes 3D. L'idée de base consiste à découper des patchs aléatoirement dans la texture source et de les coller sur la surface de sortie. La méthode profite de l'aspect stochastique des textures pour dissimuler les frontières irrégulières au système visuel humain.

Une autre approche intéressante, nommée *Chaos mosaic*, est décrite dans [GSX]. Elle possède 3 étapes principales :

- Création d'une mosaïque avec des blocs ou *tuiles* de taille fixe, pris aléatoirement dans le patch source. L'image de gauche de la figure 1.26 montre un exemple de mosaïque obtenue.
- Transformation en chaos déterministe dont les bases sont introduites dans [SJ88] : des blocs pris aléatoirement dans les tuiles sont assignés à une nouvelle position calculée

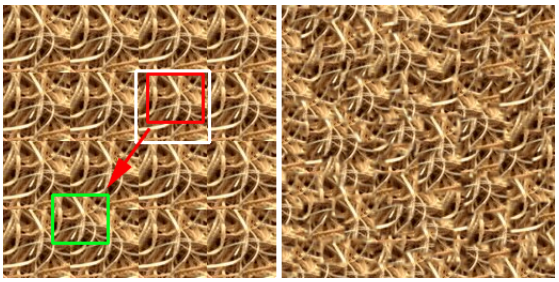


Figure 1.26 – *Illustration de la synthèse par* chaos mosaic

en utilisant la fonction *cat map* introduite dans [AA68] et définie mathématiquement par

$$(x, y) \rightarrow (2x + y, x + y) \bmod 1. \tag{1.5}$$

Cette fonction nécessite de rendre l'image torique, c'est à dire qu'elle est rendue périodique au sens du traitement des bords décrits par la figure 1.11. L'image de gauche sur la figure 1.26 montre le bloc en rouge choisi dans la tuile en blanc, qui est copié et collé sur une nouvelle position.

- Gestion des bordures : pour certaines textures stochastiques, les auteurs soutiennent que les frontières sont déjà visuellement cachées. Cependant, pour les autres types de texture, une bande de 3 pixels de large est retirée sur le bord des blocs, l'approche de [EL99] est ensuite utilisée pour remplir cette zone.

La synthèse fonctionne bien sur les textures détaillées et bruitées comme le montre l'image de droite de la figure 1.26. Afin de pallier les faibles résultats sur des textures non stochastiques, les approches suivantes évitent de recourir au placement aléatoire des patchs appliqués.

1.6.2 les méthodes exploitant le recouvrement de patchs

Les différentes approches détaillées dans cette partie construisent la texture de sortie en ajoutant des patchs qui recouvrent, dans un premier temps, la bordure de la zone déjà synthétisée. Les principales différences résident donc dans la manière de traiter cette zone de chevauchement.

Liang et al.

Le schéma décrit dans [LLX+01] ajoute des patchs sur l'image de sortie qui se chevauchent. Ainsi, il est possible se servir du chevauchement pour trouver les futurs patchs appliqués. De la même manière que dans le cas des algorithmes basés pixel, cette zone est comparée avec celles du patch source afin de sélectionner des chevauchements qui correspondent. Dans le cas contraire, il faut trouver le patch qui minimise la différence sur cette zone, puis traiter les chevauchements afin de dissimuler les frontières. Liang et al. proposent d'appliquer un fondu au niveau des chevauchements entre patchs. Les résultats contiennent typiquement des zones floues dans le cas de textures détaillées. Néanmoins, l'idée du chevauchement a été reprise dans les schémas suivants.

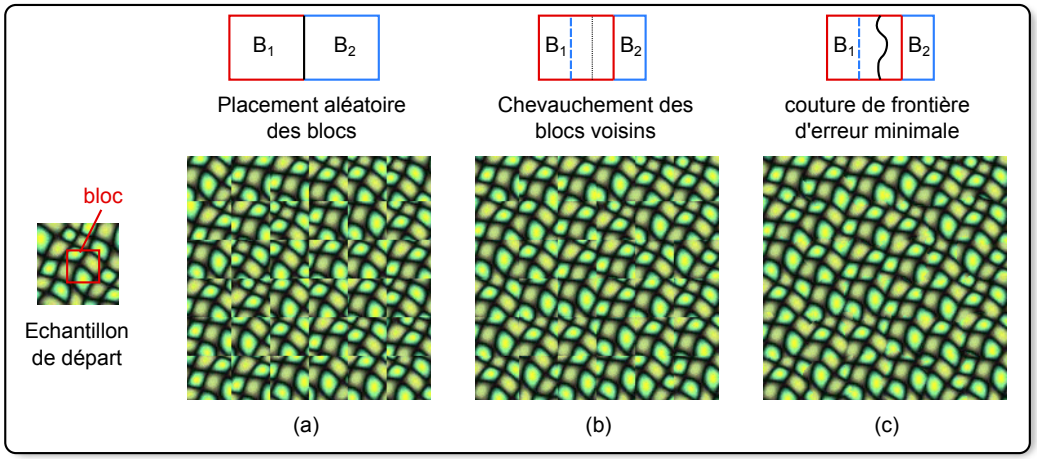


Figure 1.27 – *Synthèse par approche patch de A. Efros et W. Freeman*

Image Quilting

A.A. Efros et W.T. Freeman ont proposé une solution élégante en trois étapes distinctes développées dans [EF01], qu'ils ont appelée *Image Quilting* du terme *matelassage* utilisé dans le domaine textile. Cet article définit les bases de l'agencement et de la couture entre les patchs ajoutés qui seront repris par l'article de référence des méthodes basées patch [KSE+03]. Le modèle utilisé est encore inspiré des champs aléatoires de Markov. La figure 1.27 illustre les trois étapes principales de cet algorithme.

- Des blocs aléatoirement choisis dans le patch source sont d'abord copiés, créant ainsi la mosaïque (a).
- Dans l'ordre *raster scan*, pour chaque bloc de sortie, l'algorithme recherche dans le patch source un échantillon de blocs qui satisfont une contrainte de chevauchement avec les blocs de gauche et de dessus quand ils existent. Pour cela, une différence L2 est appliquée sur les zones de chevauchement, puis un seuil de tolérance est appliqué pour déterminer cet échantillon. Un bloc de cet échantillon est ensuite aléatoirement choisi puis copié sur l'image de sortie de l'algorithme.
- Enfin, l'erreur sur la zone de chevauchement avec ce nouveau bloc est calculée afin de déterminer le chemin qui permet de découper cette zone de manière à ce que l'erreur à la frontière soit minimale.

Malgré l'élégance de l'approche, l'association de la sélection aléatoire de blocs et de l'algorithme de couture ne permet pas toujours de trouver des assemblages heureux suivant les types de textures. Les textures lisses par exemples peuvent se voir découpées avec apparition de frontières visibles à l'œil. D'un autre côté, les approches pixel peuvent s'avérer très efficaces pour les textures lisses, notamment l'approche de Wei et Levoy présentée dans la partie 1.5.2, il serait donc judicieux de pouvoir choisir le bon algorithme en fonction du type de texture à synthétiser. L'algorithme suivant tente lui d'associer les deux méthodes dans un même schéma de synthèse.

Méthode hybride patch/pixel.

La méthode présentée dans [NA03] fait le constat du problème posé au niveau des régions de chevauchement de patch dans le schéma de [EF01]. Deux améliorations sont donc apportées faisant le lien entre approches pixels et approches patchs :

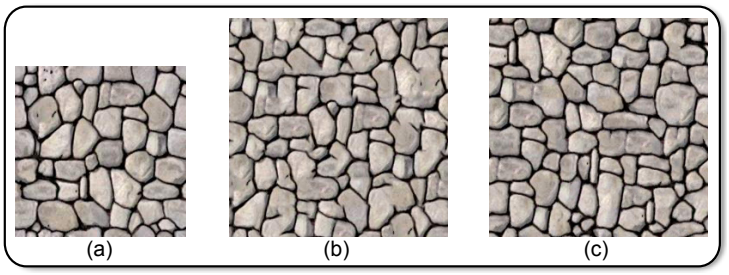


Figure 1.28 – *Comparaison entre les synthèses de [LLX+01] (b) et [NA03] (c) à partir du même patch source (a)*

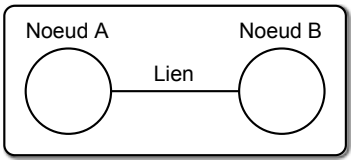


Figure 1.29 – *Maille élémentaire d'un* graphe

- Les patchs ajoutés se chevauchent, sur cette zone, les écarts entre les 2 valeurs possibles sur une position sont calculés et seuillés. Si l'erreur est trop grande, une recherche du type de celle décrite dans [WL00] est appliquée. Cette synthèse basée pixel se fait en comparant des voisinages comportant seulement des pixels avec un écart inférieur au seuil. Ils ont donc une forme définie par un masque directement corrélé au seuillage.
- Les patchs ajoutés ont une taille variable de manière à ce que la somme des erreurs précédemment définies soit bornée. Cette limite jouera le rôle de compromis entre préservation de la structure globale (ajout de patchs de grande taille), et création d'artefacts.

La figure 1.28 présente une texture pour laquelle cette technique apporte indubitablement une amélioration par rapport à l'algorithme de [LLX+01].

La méthode *Graphcut*

Cet algorithme développé par V. Kwatra et al. dans [KSE+03] fait référence pour la qualité visuelle des textures produites pour une gamme large d'échantillons sources. Il est utilisé depuis dans de nombreux travaux, notamment dans le domaine abordé dans cette thèse : la compression d'images et de vidéos. La principale contribution réside ici dans l'utilisation de la théorie des graphes présentée dans [BVZ99]. La zone de chevauchement entre la surface déjà synthétisée et le patch à ajouter est assimilée à un graphe : chaque pixel est représenté par un somment (ou nœud) ; les pixels voisins sont reliés par des arètes (liens) qui seront coupés lors de la synthèse.

Afin de produire des coutures les moins visibles possible entre les patchs, l'algorithme de *graphcut* vise à minimiser l'énergie M portée par un lien premièrement défini pour des positions adjacentes s et t par

$$M(s,t,A,B) = \|A(s) - B(s)\| + \|A(t) - B(t)\| \tag{1.6}$$

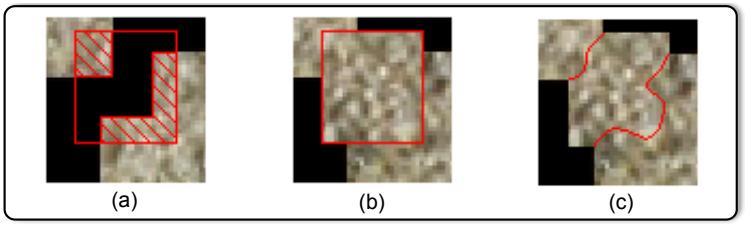


Figure 1.30 – *Processus d'ajout d'un patch*

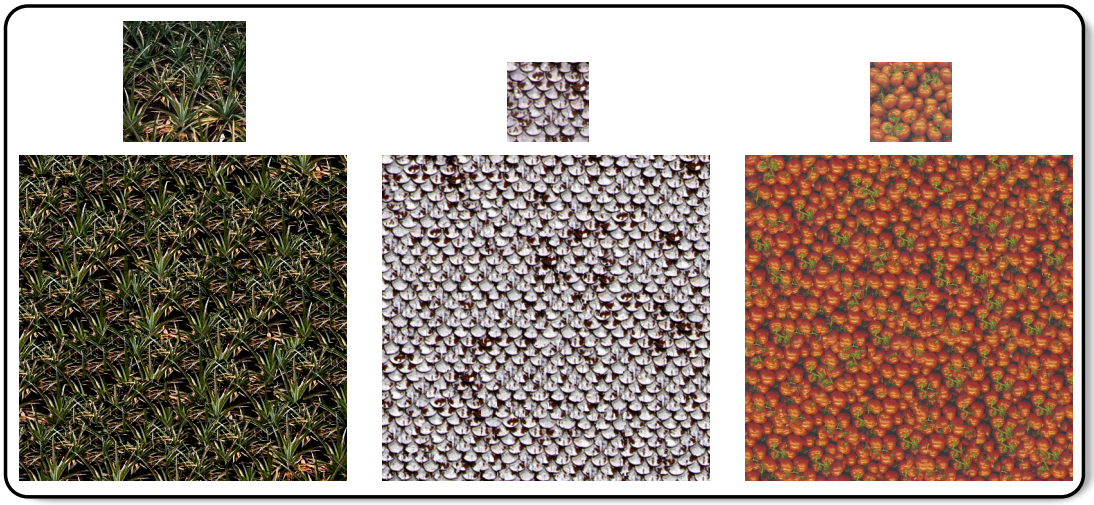


Figure 1.31 – *Quelques résultats en images de la méthode présentée dans [KSE+ 03]*

avec $A(s)$ (resp. $B(s)$) représentant la valeur du pixel à la position s appartenant au patch A (resp B).

La figure 1.31 montre des résultats en images pour trois échantillons de textures favorables à la synthèse par patchs. Il reste néanmoins certains types de textures lissées, comme celles illustrées 1.32, pour lesquelles les résultats sont perfectibles. De plus, dans certains contextes nécessitant des petites tailles de *patchs* ajoutés, le faible choix de *chemins* pour la couture, mène à l'apparition d'artefacts.

Une optimisation de cet algorithme permet la rotation de {-90 °, +90 °, 180 °} ainsi que le retournement type *miroir* des patchs à ajouter à la texture synthétisée. La figure 1.33 illustre l'avantage d'une telle technique pour les textures dont la distribution de la direction et le sens des motifs sont primordiaux dans la qualité de la synthèse.

1.6.3 La synthèse de textures quasi-régulières

La classification des textures présentée dans la figure 1.2 provenant de [Lin05] est utilisée par l'auteur pour délimiter le champ spécifique des textures quasi-régulières ou NRT pour Near-Regular Textures sur lesquelles ses travaux sont focalisés. Les premiers travaux soulignant le besoin de conserver la régularité de ces NRT lors de leur manipulation ou synthèse, sont présentés dans [LTL02]. Ces travaux partent du constat que la synthèse réussie des NRT dans la littérature, notamment des algorithmes basés pixel et patch, est uniquement due au choix judicieux des tailles et formes des voisinages ou patchs mis en

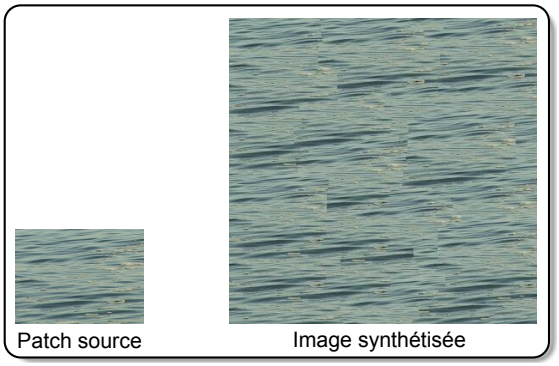


Figure 1.32 – *Synthèse défaillante pour la synthèse de fluide*

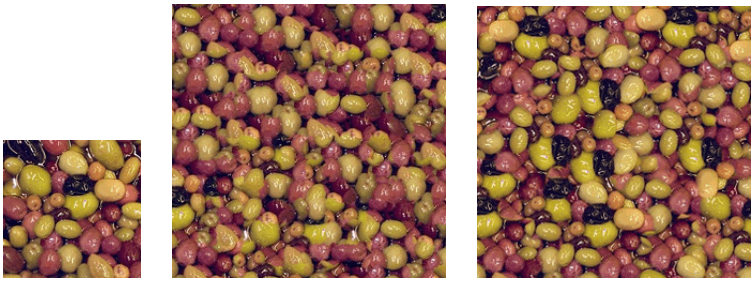


Figure 1.33 – *Comparaison des synthèses de [EF01] et [KSE+03] avec l'optimisation permettant la rotation et le retournement des patchs ajoutés.*

jeu, alors que cette approche s'appuie sur une analyse de la structure permettant d'adapter la synthèse à la texture. Ainsi, l'algorithme présenté dans [LTL02] contient deux étapes principales :

- Analyse : en se focalisant sur le patch source, cette étape détermine les symétries en translation et ses enchevêtrements. Elle trouve ainsi les *tuiles minimales* élémentaires, c'est à dire des *losanges* de taille minimale qui, apposés sur le patch source ne laissent aucun espace vide, ni se chevauchent. La figure 1.34 illustre ces *losanges* en jaune sur les images (a) et (b). Pour chaque tuile ainsi calculée est ensuite construite la *tuile maximum* correspondante, qui est un rectangle dans lequel est inscrit le *losange* et se chevauche avec quatre de ses *tuiles maximum* voisines.
- Synthèse : pour chaque nouveau point clé de la surface à synthétiser, cette étape choisit aléatoirement une *tuile maximum* dans le patch source et la centre en ce point. Un dégradé inspiré de [SS00] est ensuite appliqué sur les zones de chevauchement afin de lisser les transitions entre les patchs.

Un travail intéressant de comparaison a par ailleurs été mené dans [LHW+06] en collaboration avec d'autres auteurs sur la synthèse de textures quasi-régulières.

La partie suivante présente une nouvelle classe de synthétiseurs visant à faire converger une fonction d'énergie calculée sur la surface synthétisée.

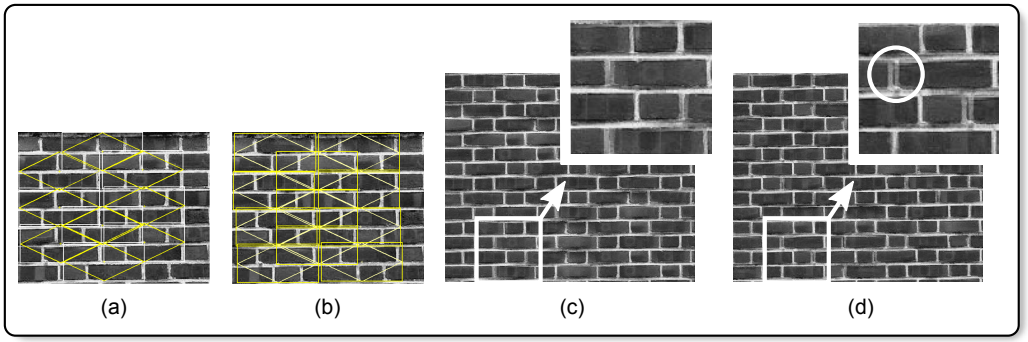


Figure 1.34 – *(a) and (b) montrent deux positions différentes de croisillons. (c) présente un résultat de synthèse et (d) un cas où la synthèse est défaillante due à une mauvaise correspondance.*

1.7 La synthèse EM

La synthèse EM pour Espérance - Maximisation est une famille de méthodes itératives permettant de faire converger la surface synthétisée vers une images ayant les propriétés de l'échantillon source.

1.7.1 De la théorie EM à la synthèse de texture

L'un des premiers papiers traitant l'estimation du maximum de vraisemblance d'une population, dans lequel toutes les données ne sont pas disponibles, est présenté dans [Har58]. C'est cependant une approche dédiée à quatre exemples traités qui est développée. Le but n'est pas ici de synthétiser une large gamme de texture mais d'avoir un algorithme performent pour 4 types de textures connus. Une étude générique de la résolution de ce problème en utilisant la méthode EM a été introduite par la suite dans [dLR77]. Un livre de G. J. McLachlan et T. Krishnan [MK97] pose les bases de la théorie EM ainsi que des extensions possibles. Il permet, avec l'article [dLR77], de décrire le cheminement de l'algorithme fondamental à la synthèse de texture EM vue par V. Kwatra dans [KAK05].

Partons de deux espaces $\mathcal{X}$ et $\mathcal{Z}$ liés par une fonction surjective régie par

$$\begin{array}{c} x \to z(x) \\ \mathcal{X} \to \mathcal{Z} \end{array} \tag{1.7}$$

Dans cette fonction, z représente les données observées, *i.e.* une version vectorisée de l'ensemble des voisinages du patch $\mathcal{Z}$ dans le cas de la synthèse. Les vecteurs x dans $\mathcal{X}$ représentent les données complètes, *i.e.* les vecteurs voisinages de l'image à synthétiser qui seront regardées à travers z. En effet, chaque voisinage considéré x_p de l'image de sortie est observé à travers son correspondant z_p via le paramètre Φ_p qui les relie. La figure 1.35 illustre le lien entre les différents voisinages représentés par les carrés noirs. L'équation 1.7 définit une surjection ; aussi, deux voisinages dans l'image de sortie peuvent avoir le même correspondant dans le patch. L'échantillon x suit la loi $f(x|\Phi)$ dépendant du vecteur Φ des paramètres inconnus, ainsi que la loi correspondante $g(z|\Phi)$ pour les densités de l'échantillon observé. La relation reliant ces deux fonctions est donnée par

$$g(z|\Phi) = \int_{\mathcal{X}(z)} f(x|\Phi)dx. \tag{1.8}$$

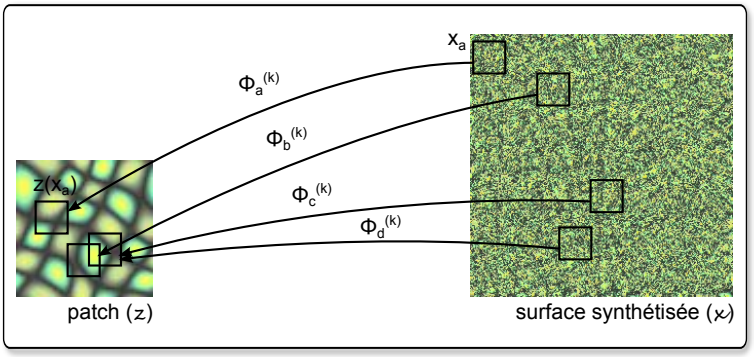


Figure 1.35 – *Illustration des paramètres de la théorie EM au contexte de la synthèse.*

Ainsi, on appelle la *log-vraisemblance*, pour l'échantillon x, la fonction

$$L(x;\Phi) = log(f(x,\Phi))). \tag{1.9}$$

L'algorithme EM vise donc à déterminer le vecteur de paramètres Φ qui va maximiser cette log-vraisemblance. Les deux étapes fondamentales nécessitent donc de procéder comme suit, à l'itération k :

- étape E : calcul de l'espérance Q par

$$Q(\Phi,\Phi^{(k)}) = E_{\Phi^{(k)}}\{L(\Phi)|z\}; \tag{1.10}$$

- l'étape M consiste à déterminer $\Phi^{(k+1)}$ solution du système d'équations

$$M(\Phi^{(k)}) = \underset{\Phi}{\operatorname{argmax}}\, Q(\Phi,\Phi^{(k)}). \tag{1.11}$$

L'étape M consiste donc pour la synthèse à maximiser la ressemblance entre x et y. Dans ce cadre, Φ correspond au paramétrage du transfert de la texture vers x, qui est opéré en comparant des voisinages de pixels de l'image de sortie avec ceux du patch. C'est donc un vecteur de paramètres définissant la position du meilleur voisinage trouvé dans le patch pour chaque position observée dans l'image de sortie.

Les étapes E et M sont ainsi alternées jusqu'à ce que

$$L(\Phi^{(k+1)}) - L(\Phi^{(k)}) < Th \tag{1.12}$$

avec Th le seuil fixé pour déterminer l'itération à partir de laquelle on considère qu'il y a convergence.

La partie suivante décrit un algorithme s'inspirant de cette théorie, afin de faire converger une surface synthétisée vers une vraisemblance maximum entre les vecteurs voisinages pris sur cette surface et ceux du patch.

1.7.2 Algorithme de référence par Kwatra et al.

Cet algorithme, développé dans [KAK05], suit les propriétés des champs de Markov, comme la plupart de ceux décrits précédemment. La valeur d'un pixel dépend donc uniquement du voisinage qui l'entoure, cette relation étant indépendante de la position du pixel sur la surface synthétisée. Soient $\mathcal{X}$ et $\mathcal{Z}$ correspondant toujours à l'image synthétisée

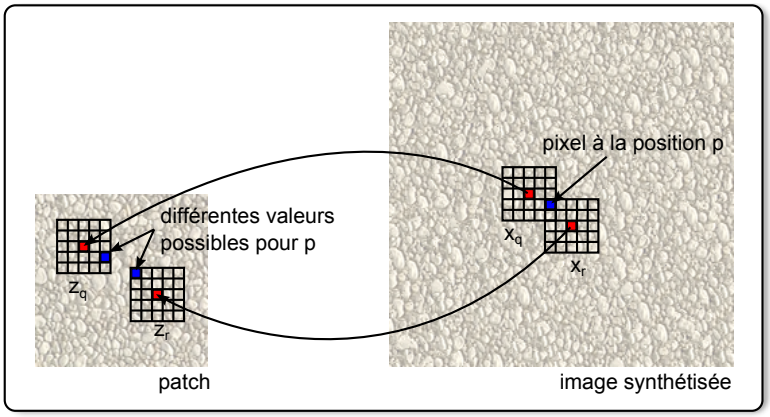


Figure 1.36 – *Synthèse EM : illustration de l'étape E.*

et au patch respectivement, x et z leurs versions vectorisées. On appelle $\mathcal{V}_p$ les voisinages centrés en x_p de l'image synthétisée et z_p la position dans le patch du voisinage qui lui ressemble le plus suivant la norme Euclidienne. Ainsi, la vraisemblance à maximiser passe maintenant par la minimisation d'une énergie définie par

$$E(x;\{z_p\}) = \sum_{p\in X^+} \|x_p - z_p\|^2 , \tag{1.13}$$

où X^+ représente un sous-échantillon de X tel que les voisinages considérés se chevauchent suffisamment. Les auteurs argumentent que le calcul sur tous les voisinages de l'image de sortie, en plus d'être coûteux, est redondant. L'étape M consiste donc à trouver les meilleurs représentants dans le patch des voisinages centrés sur l'ensemble des pixels inclus dans X^+ au même sens que dans les algorithmes de [EL99] et [WL00].

Les voisinages $\mathcal{V}$ se chevauchant, la figure 1.36 illustre le fait que voir l'échantillon à travers les meilleurs représentants z conduit à résoudre l'étape E. Cette étape vise à calculer l'espérance du pixel à la position p, de façon à minimiser les contraintes entre les valeurs possibles suivant z_p et z_q. Minimiser cette énergie sur toute l'image à synthétiser revient à résoudre le système d'équations

$$\frac{\partial E(x;\{z_p\})}{\partial x} = 0. \tag{1.14}$$

Afin de minimiser cette énergie, V. Kwatra choisit d'assigner à la position p la moyenne des valeurs qu'il aurait eues à partir de chaque position relative des voisinages qui se chevauchent sur sa position. Il s'écarte ainsi de la théorie EM fondée sur la maximisation de la log-vraisemblance d'une loi de densité de probabilité, qui nécessiterait en plus de calculer la covariance sur x.

L'algorithme de base procède ainsi comme suit :

1. initialisation aléatoire des z_p^0 dans le patch $\mathcal{Z}$ pour tous les pixels dans X^+,
2. pour chaque itération :
   - E-step : à chaque pixel de l'image de sortie, on affecte la valeur moyenne des valeurs possibles suivant la provenance des voisinages qui se chevauchent sur lui. Cette étape est illustrée par la figure 1.36.
   - M-step : recherche des meilleurs représentants z_p dans le patch pour chaque x_p du sous-échantillon X^+ dans l'image de sortie.

3. répéter 2 jusqu'à l'une des conditions suivantes :
   - les z_p à l'itération n+1 ne changent plus de position.
   - $E(x;\{z_p\}) < Th$ fixé par l'utilisateur.

Le pseudo-code de l'algorithme 1.2 donne un aperçu du processus.

Algorithme 1.2: Algorithme de synthèse EM par V. Kwatra

Entrées : Patch source Z, dimensions de l'image de sortie
Sorties : Texture synthétisée X

```
Initialisation : z_p^0 ← voisinage aléatoire de Z, ∀p ∈ X^+;
pour n = 0 : N faire
    x^{n+1} ← argmin_x E_t(x;{z_p^n}) ; // étape E
    z_p^{n+1} ← meilleur correspondant de x_p^{n+1} dans Z ∀p ∈ X^+ ; // étape M
    si z_p^{n+1} = z_p^n ∀p ∈ X^+ alors
        x ← x^{n+1};
        arrêt;
    fin
fin
```

Comme dans [WL00], cette approche est confrontée à la lenteur de l'exécution de la recherche des meilleurs représentants dans le patch pour l'étape M. Même problème, même solution : pour accélérer le schéma, la recherche est exécutée en traversant un arbre pré-calcul structurant les vecteurs voisinages du patch en sous-classes.

De même, une approche *multirésolution* est proposée, multirésolution ayant deux acceptions :

- l'utilisation d'une pyramide Gaussienne
- la variation de la taille des voisinages suivant les itérations.

Le but de cette double manœuvre est de permettre à l'algorithme de capter les structures et les larges motifs, puis de raffiner les détails aux résolutions supérieures et en considérant des voisinages plus petits mais aussi plus denses puisque la séparation entre voisinages reste de $Taille_{voisinage}/4$. L'approche multirésolution n'a pas ici la même signification qu'en 1.5.2 du fait que le voisinage considéré n'est pas localisé sur plusieurs étages de la pyramide. En effet, à chaque niveau, plusieurs itérations sont appliquées. L'image est ensuite sur-échantillonnée pour le passage à un étage inférieur de la pyramide de sortie. V. Kwatra s'est par ailleurs servi des bonnes propriétés de réarrangement de la texture de sortie dans des travaux présentés dans [KAK+07] permettant l'extension de cet algorithme pour la synthèse de fluides en 3 trois dimensions.

1.7.3 Version utilisant la k-cohérence

Cette version EM de la synthèse de texture est décrite dans [HTW07]. Cet algorithme, qui reprend largement l'approche de [KAK05], propose d'utiliser la k-cohérence pour optimiser la recherche du meilleur pixel dans le patch,) chaque itération. L'algorithme inclut donc aussi l'étape de prétraitement afin de déterminer les pixels cohérents dans le patch source. Aussi, comme dans [TZL+02], l'approche nécessite de stocker d'où provient chaque pixel de l'image synthétisée, alors que l'approche de [KAK05] ne nécessitait de stocker que la provenance des meilleurs voisinages pour le sous-échantillon de pixels X^*. L'approche de k-cohérence va, par conséquent, modifier aussi l'étape E puisqu'elle nécessite de stocker la provenance des pixels de sortie. En effet, l'assignation de la moyenne décrite dans la partie 1.7.2 n'est donc plus possible. L'algoithme comporte maintenant les étapes suivantes :

1. calcul de la moyenne de la même manière.
2. recherche de la valeur des pixels de patch pris en compte qui minimise l'écart à la moyenne.
3. assignation de sa valeur dans l'image de sortie et stockage de sa provenance dans le patch.

Les auteurs prétendent que cette modification, en plus d'accélérer considérablement la recherche lors de l'étape M, permet d'éviter le flou introduit par la moyenne affectée au pixel courant dans l'algorithme de [KAK05]. Le pseudo code donné dans l'algorithme 1.3 décrit les différents processus, il reprend les notations introduites par l'algorithme 1.2. Cette approche introduit néanmoins quelques contraintes, liées à la k-cohérence. Chaque pixel de sortie est lié à son correspondant dans le patch, et c'est finalement l'ensemble de ces liens qui est mis à jour tout au long de la synthèse. L'approche multirésolution requiert donc de sur-échantillonner ces tableaux, empêchant ainsi une interpolation comme proposé dans [KSE+03] ou un filtrage adéquat tel le filtrage Gaussien utilisé dans d'autres approches multirésolutions.

La section suivante propose de détailler un outil atypique, qui permet en théorie, de construire un taille de patch réduite, à partir d'une large surface. Ce patch est sensé contenir le maximum d'informations, en vue de reconstruire ensuite une texture proche de la surface initiale. Les auteurs ont donc logiquement appelé cet outils : synthèse inverse de texture.

1.8 Synthèse inverse de texture

On appelle synthèse inverse la technique qui consiste à construire le patch optimal afin de recréer une surface connue au départ. Ces travaux présentés dans [WHZ+08] permettent donc le contraire des algorithmes présentés précédemment, c'est à dire construire une petite image de texture à partir d'une grande. Ce concentré doit donc posséder et regrouper les motifs et les variations contenues dans la grande texture de départ, afin de permettre à la synthèse, ensuite, de recréer une texture visuellement proche de l'originale. La figure 1.37 illustre l'avantage de cette technique pour une texture comportant un motif de premier plan devant un fond globalement variant du noir à l'orange. La synthèse produite à partir de l'algorithme EM et d'un patch découpé dans l'image source ne permet pas de recréer la variation de couleur du fond alors que la synthèse inverse recrée une image visuellement proche de la source. Malgré le fait que la reconstruction de cet exemple nécessite un champ d'orientation guidant les variations globales, il reste que cette technique est intéressante pour construire un *patch* de taille réduite possédant un maximum d'informations à propos des primitives et de leurs variations. Techniquement, l'énergie à minimiser diffère de 1.13 par l'ajout d'un terme inverse, soit maintenant :

$$E(x;z;w) = \frac{1}{|X^+|}\sum_{p\in X^+}|x_p(w_p)-z_p|^2 + \frac{\alpha}{|Z^+|}\sum_{q\in Z^+}|x_q(w_q)-z_q|^2\,. \qquad (1.15)$$

Dans cette équation, le terme w représente un champ d'orientation qui permet de contraindre les variations globale de la synthèse. Le terme α a été empiriquement déterminé à $\alpha = 0.01$ par les auteurs. L'algorithme 1.3 permet de décrire l'imbrication de cette technique avec les algorithmes EM précédemment étudiés. Les auteurs utilisent particulièrement l'algorithme de k-cohérence afin de garder constamment un lien entre les pixels sources et les pixels dans l'image construite, ce lien étant nécessaire dans les allers et retours entre minimisation des termes de synthèse *avant* et *inverse*. Cette solution paraît spécialement adéquate

Algorithme 1.3: Algorithme de synthèse inverse.

Entrées : Image de départ X, Champ de départ W
Sorties : Patch Z

Initialisation :
$z_p^0 \leftarrow$ voisinage aléatoire dans Z $\forall p \in X^+$;
$x_q^0 \leftarrow$ voisinage aléatoire dans X $\forall q \in Z^+$;
$w^0 \leftarrow$ initialisation;
pour *l'itération* $m = 0 : M$ **faire**
  pour *la résolution* $l = 0 : L$ **faire**
    si $l < L$ **alors** $w \leftarrow$ sous-échantillonnage au niveau L
    si $l > 0$ **alors** $z \leftarrow$ sur-échantillonnage au niveau $l - 1$
    pour *l'itération* $n = 0 : N$ **faire**
      $z^{n+1} \leftarrow \text{argmin}_z E(x, y, w)$; // *étape E*
      pour chaque $p \in X^+$ **faire**
        $z_p^{n+1} \leftarrow \text{argmin}_{z_p} \|x_p(w_p - z_p)\|^2$; //*étape M inverse*
      fin
      pour chaque $q \in Z^+$ **faire**
        $x_q^{n+1} \leftarrow \text{argmin}_{x_q} \|x_q(w_p - z_q)\|^2$; //*étape M avant*
      fin
      si $z_p^{n+1} = z_p^n$ $\forall p \in X^+$ **alors**
        $z \leftarrow z^{n+1}$;
        si l==L **alors**
          $w^{n+1} \leftarrow \text{argmin}_w E(x, y, w)$; // *étape E pour w*
          si $w^{n+1} == w^n$ **alors**
            $w \leftarrow z^{w+1}$;
            arrêt;
          fin
        fin
      fin
    fin
  fin
fin

pour recréer une texture que l'on voudrait compresser en un patch de taille réduite. C'est cependant dans des contextes restreints de reconstruction d'une texture variant avec les paramètres w que cette méthode donne des résultats satisfaisant. Il apparaît compliqué d'utiliser de telles techniques dans un contexte de compression avec des contraintes fortes aux bords et des variations difficilement modélisables par la carte w, qui serait par ailleurs à compresser.

La partie suivante vise à donner un aperçu des techniques *d'inpainting*. Ces méthodes permettant de prolonger du signal qui n'est pas forcément de la texture au sens défini précédemment, cette partie dressera l'état de l'art des méthodes les plus susceptibles de se rapprocher du contexte de ces travaux de thèse.

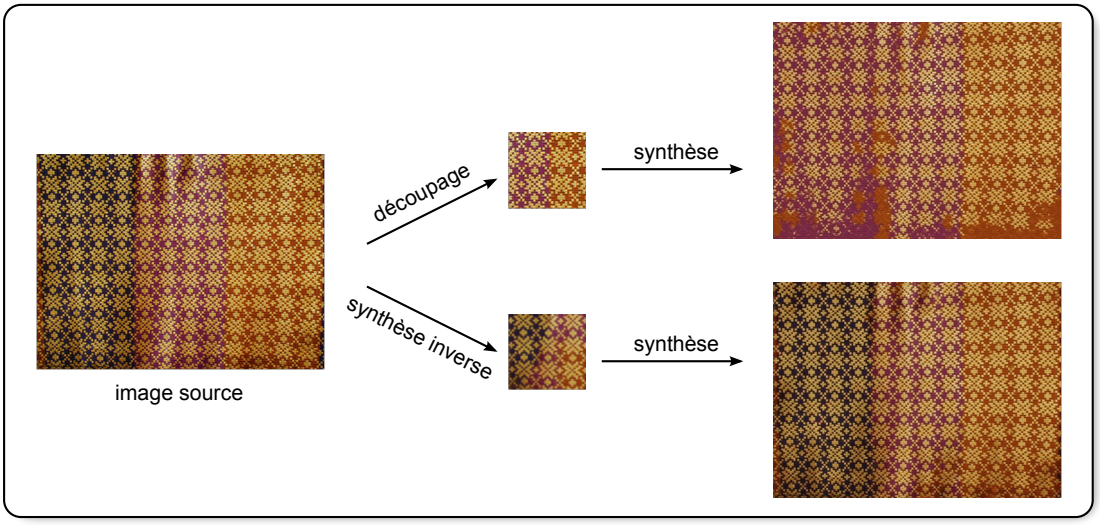


Figure 1.37 – *Synthèse inverse de texture, résultats avec découpage d'un* patch *en haut et avec la création du* patch *par synthèse inverse en bas.*

1.9 Méthodes d'*Inpainting*

On entend par *inpainting* les méthodes qui consistent à reboucher une zone perdue ou enlevée d'image à partir des informations contenues dans son voisinage. Ainsi, comme le suggère le terme, il s'agit de propager les formes et les textures spatialement comme un peintre qui continuerait un tableau auquel il manquerait une région. Deux principales différences sont à noter par rapport à la synthèse de texture telle qu'elle est présentée dans ce chapitre :

- Prise en compte de contraintes de bord : il faut que la transition entre les pixels connus et la région soit visuellement imperceptible.
- Possibilité de structures présentes ou découpées : en plus de la texture pseudo-stationnaire, il faut donc propager les contours de manière à ce que la reconstruction apparaisse naturelle à l'œil. C'est le cas lors de la restauration de contenu ou de remplissage d'une région sur laquelle il y avait un logo d'incrusté par exemple.

Les méthodes d'inpainting sont donc principalement utilisées pour reboucher des zones de tailles réduites lors de restauration de contenu altéré ou perdu pendant la transmission. Contrairement aux travaux de cette thèse, focalisés sur la synthèse de régions texturées pseudo-stationnaires, ces algorithmes s'attachent donc à propager des structures. Cependant les liens étroits entretenus entre ces approches nous amènent à évoquer les principales techniques d'inpainting. En effet, ces deux méthodes peuvent s'avérer bilatéralement complémentaires :

- Synthèse de texture le long de structures propagées par une méthode d'inpainting.
- Utilisation de techniques d'inpainting entre deux régions synthétisées.

1.9.1 Propagation de signal anisotrope

L'un des premiers travaux visant à résoudre de manière automatique un problème d'*inpainting* est décrit dans [BSCB00]. Le système est fondé sur la propagation anisotrope de signal, c'est à dire que celle-ci se fait uniquement dans certaines directions, permettant de propager des formes géométriques. Cette propagation se fait en résolvant un système

Figure 1.38 – *Décomposition des images pour l'inpainting des zones blanches sur l'image source* Barbara *par l'algorithme présenté dans [BVSO03].*

d'équations aux dérivées partielles (EDP). Les travaux de Chan et Shen reprennent l'idée de la diffusion anisotrope en introduisant un modèle de *Variation Totale* fondé sur la résolution d'un problème d'Euler-Lagrange [CS00]. Un schéma plus abouti a été ensuite proposé dans [CS01] propagent l'information dans des zones plus étendues.

Une méthode plus efficace en termes de coûts de calculs est présentée dans [OBMC01]. Elle reste cependant limitée à la reconstruction d'une zone très étroite, puisque fondée sur une diffusion isotrope du signal. De plus, son efficacité dépend d'une information supplémentaire, fournie par l'utilisateur, qui détermine par des segments l'arrêt de la diffusion isotrope à l'origine de l'introduction de flou, lors de la reconstruction de contours.

Un algorithme itératif EM est décrit dans [FS05]. Contrairement à l'algorithme utilisé dans la synthèse de texture présentée dans la partie 1.7.2, l'espérance repose sur une approche utilisant les représentations parcimonieuses. Ces dernières consistent en la décomposition du signal sur une base appelée dictionnaire ayant un nombre d'atomes très supérieur à la dimension du signal. Cette décomposition introduit un grand nombre de valeurs nulles. L'étape M correspond ici à la décomposition la plus parcimonieuse alors que l'étape E correspond à la mise à jour de l'estimation en fonction de la nouvelle distribution.

La propagation du signal est efficace pour les formes ou objets coupés dans la zone à reproduire, mais il faut aussi traiter les textures, plus difficiles à prédire. La partie suivante propose de décrire quelques méthodes qui intègrent, en plus des techniques précédemment abordées, la synthèse de texture pour former un schéma complet permettant de reboucher des régions perdues à l'aide de leur environnement.

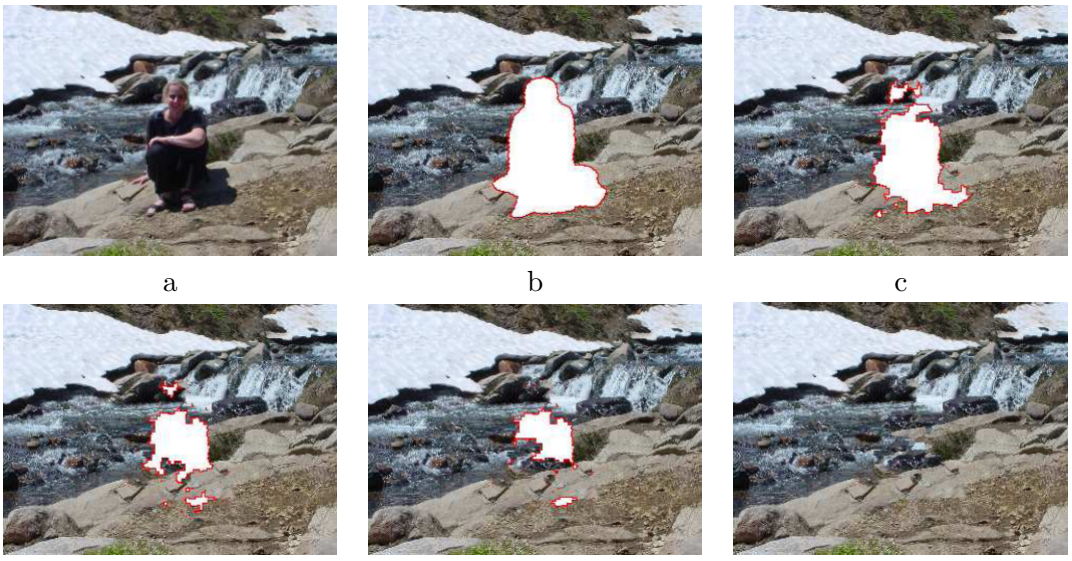


Figure 1.39 – *Résultats intermédiaires de l'*inpainting *de [CPT04], suivant l'ordre de construction.*

1.9.2 Combinaison de synthèse de texture et propagation géométrique

Les équations à dérivées partielles permettent la propagation de contours mais sont insuffisantes pour propager des motifs de textures à mesure que la surface est large. Avec l'évolution rapide, en parallèle, des algorithmes de synthèse de texture, des approches mixtes d'*inpainting* ont vu le jour, alliant synthèse de texture et propagation géométrique. La partie complexe de ce processus est alors de décorréler propagation de structure et ajout des informations de texture. L'approche présentée dans [BVSO03], propose d'extraire les structures via un algorithme de minimisation de la variation totale [VO03]. Ainsi le signal image contient une version de *structure* S et une version texture T

$$I = S + T \tag{1.16}$$

Ensuite, S est remplie en utilisant la méthode décrite dans [BSCB00], alors que T est traitée en utilisant l'algorithme de [EL99] présenté en section 1.5.1. La figure 1.38, provenant de [BVSO03], montre la reconstruction de zones enlevées en utilisant cette décomposition du signal. L'image en haut à gauche montre la source et les zones retirées, la seconde ligne montre la décomposition en une image de contours et une image de texture, enfin, la dernière ligne montre les reconstructions de chaque image. L'image finale, résultant de la somme de la dernière ligne est présentée en haut à droite. Les deux approches utilisées sont :

- la synthèse de texture de [EL99], présentée en section 1.5.1 ;
- la méthode de propagation de contours proposée dans [BSCB00]

La deuxième approche faisant référence dans l'utilisation couplée de synthèse de texture et des techniques premières d'*inpainting* est décrite dans [CPT04]. Contrairement aux approches utilisant des équations de diffusion, la propagation de structure s'inspire d'algorithmes de synthèse de texture et notamment [EL99]. En effet, la valeur d'un nouveau pixel dépend de la recherche d'un pixel candidat dans le signal connu, en comparant leur voisinage. La propagation de la structure de cette manière est assurée par l'ordre de la

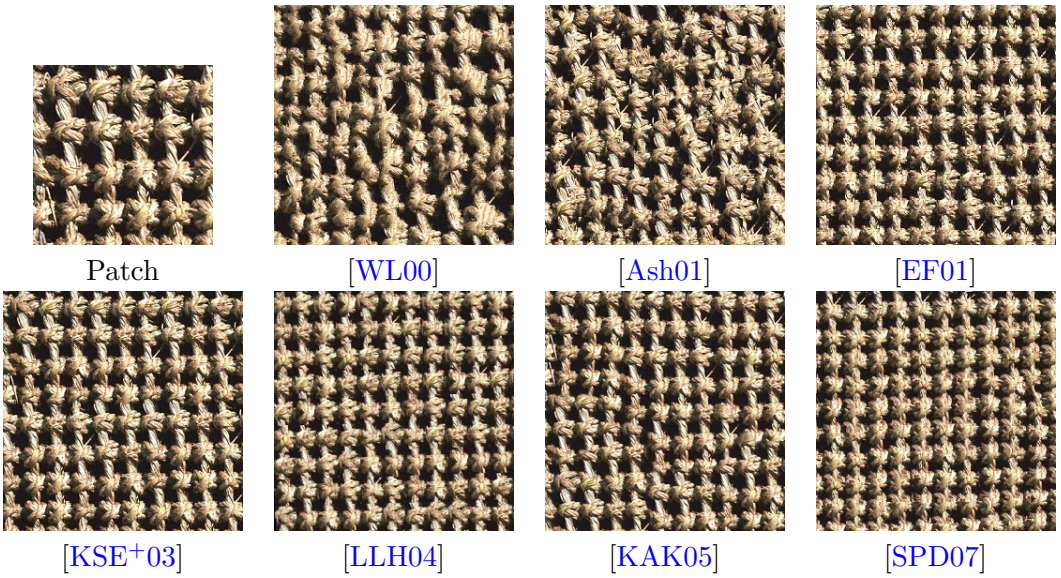


Figure 1.40 – *Comparaison des différentes synthèses sur l'échantillon de texture 9 de la base VisTex [PGM+95]. Les résultats des algorithmes de [WL00], [Ash01] et [KAK05] sont issus d'une implémentation personnelle alors que les autres proviennent directement de la littérature.*

synthèse qui prend en compte les gradients présents sur les bords de la région manquante. Une carte de confiance est mise en place afin de gérer l'ordre de construction des pixels manquants. Cette carte favorisant les zones à fort gradient, les contours sont d'abord propagés, la mise à jour de la carte permet ensuite de construire le reste de la zone manquante dans un ordre logique, géométriquement. La figure 1.39 présente les résultats en cours de synthèse afin d'illustrer l'ordre de l'*inpainting* et la qualité du résultat final.

Kumar et al. proposent dans [KBBN05] de reprendre l'*inpainting* décrit dans [CPT04] pour en apporter deux extensions : la prise en compte de la dimension temporelle pour l'inpainting de séquences vidéo ainsi qu'une recherche non paramétrique des meilleurs patchs dans le signal existant, exécutée dans le domaine de Fourier [KDM02].

Plusieurs approches ont ensuite été présentées dans le même registre. La solution décrite dans [YHS] propose par exemple de décomposer les hautes fréquences assimilées à la texture et les basses fréquences via le domaine DCT. La synthèse de texture multirésolution présentée dans [WL00] et décrite à la section 1.5.2 est utilisée pour la synthèse des hautes fréquences.

1.10 Comparaisons et choix

Cette section vise à montrer et à comparer les résultats visuels pour certains algorithmes développés précédemment. Certains algorithmes ont été implantés, principalement les méthodes *pixel* et *patch* afin de pouvoir choisir les méthodes adaptées au schéma de compression souhaité. Quelques textures provenant de la base de donnée de [PGM+95] sont heureusement largement utilisées par les auteurs pour montrer l'efficacité de leur synthétiseur. Le fait de comparer certains résultats provenant d'une implémentation personnelle avec ceux de la littérature introduit inévitablement un biais du fait qu'il faille paramétrer les algorithmes. Toutefois, une application toute particulière a été portée pour montrer les résultats avec les meilleurs paramètres trouvés, afin de ne pas léser

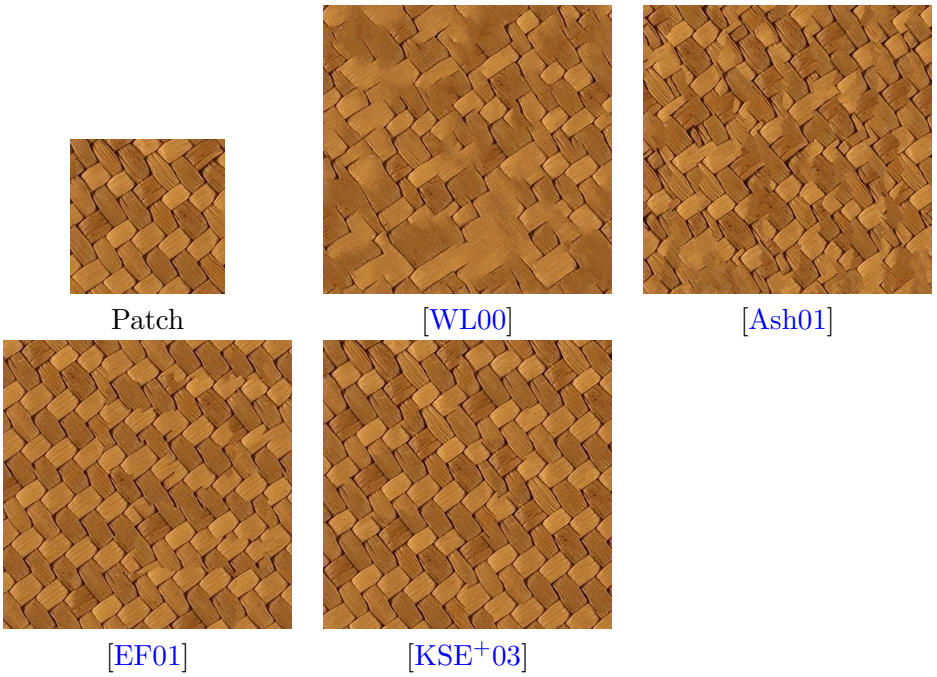


Figure 1.41 – *Comparaison des différentes synthèses sur l'échantillon de texture 7 VisTex [PGM+95]. Les résultats des algorithmes de [WL00], [Ash01] et [KSE+03] résultent d'une implémentation personnelle alors que les autres proviennent directement de la littérature*

les algorithmes implémentés par rapport aux résultats de la littérature, qui sont eux aussi, logiquement optimisés. Il est par ailleurs facile de démontrer les failles d'algorithmes implémentés en modifiant certains paramètres. Les figures 1.40 et 1.41 montrent les résultats de différentes approches étudiées dans ce manuscrit. On s'aperçoit des textures structurées avec des primitives bien définies, les méthodes *patch* donnent des résultats satisfaisants.

Bien que les choix des algorithmes retenus soient détaillés dans les parties contributions, citons quelques caractéristiques essentielles à prendre en compte. Avec la qualité des résultats visuels, le critère fondamental dans le choix d'utilisation d'un algorithme réside dans son adaptabilité au contexte. Dans celui de la synthèse ou du raffinement de textures enlevées ou altérées dans les vidéos, plusieurs contraintes seront à prendre en compte :

- les contraintes aux bords des régions,
- les contraintes à la forme et à la taille du patch,
- les contraintes de paramétrage, et adaptation à une gamme de textures la plus large possible,
- les contraintes temporelles...

La liste n'est pas exhaustive mais permet de se rendre compte que les algorithmes rigides en termes de paramètres, itératifs ou encore dédiés à un type restreint de textures seront donc à proscrire.

1.11 Conclusion

La présentation des différentes approches de synthèse de texture dans ce chapitre montre la difficulté de les trier d'une part, et de les comparer d'autre part. En effet, les différences dans les approches se situent au concept même de la modélisation choisie :

définir une *distribution probabiliste* de la surface à synthétiser, construire la texture de manière *particulaire*, motif à motif, ou pixel à pixel, itérer un processus convergeant vers une distribution maximisant l'espérance attendue du signal synthétisé... Le comportement des algorithmes est de plus directement circonstancié au type de texture auquel ils s'attaquent. Il faut néanmoins faire des choix pour le contexte de la compression d'images fixes et animées. Un des points fondamentaux réside dans la facilité de paramétrer l'algorithme. En effet, l'automatisation du système de compression nécessite un nombre minimum de paramètres à définir d'une part, mais aussi de pouvoir les adapter en fonction de la région texturée à traiter. Les approches non-paramétriques, construisant la surface synthétisée pixel à pixel ou par groupes de pixels, apparaissent donc les méthodes les plus aisées à mettre en place. C'est ce qu'il ressort notamment des approches présentées dans le chapitre suivant qui visent à exploiter ce type d'algorithmes pour la reconstruction de régions texturées au décodeur.

CHAPITRE 2

La compression vidéo.

Le format de télévision numérique standard, tel qu'on le reçoit au format SD (*Standard Definition*) en Europe, comporte 720×576 pixels à la fréquence de 25 images par seconde. Afin d'approcher la qualité de son prédécesseur analogique, en considérant que 12 bits sont nécessaires pour représenter la couleur d'un pixel, il faudrait transmettre une quantité d'information de 124.4 Mbits par seconde. Or un canal TNT, large de 8 MHz, a un débit numérique limité à 40 Mbits/s. La diffusion de formats Haute Définition (1920×1080) sans compression paraît donc compromise.

Heureusement, une multitude de redondances sont contenues dans les images et encore plus dans les vidéos. Aussi, Claude Shannon a défini le terme d'**entropie** comme la quantité d'information réellement contenue dans un signal source. Dans le cas d'un système de communication d'un signal numérique entre un émetteur et un récepteur, l'entropie de la source d'information correspond à l'incertitude du récepteur par rapport à ce que la source va transmettre. Dans le cas du codage réversible, *i.e.* sans perte d'informations, le but de la compression vidéo consiste à diminuer la quantité de données à transmettre au récepteur en supprimant le maximum de redondances, ceci afin que la quantité d'informations transmise tende vers l'entropie du signal source. Si le taux de compression obtenu ne suffit pas, le codage avec pertes permet d'accroître ce taux, au prix d'une dégradation du signal. C'est le **codeur** de l'émetteur qui se chargera de transformer l'image ou la séquence vidéo en un **train binaire** dans lequel l'information source est condensée. Le **décodeur** du récepteur se chargera d'analyser le signal reçu pour reconstruire les images ou les vidéos.

Ce chapitre a pour but de présenter les enjeux et les contraintes qui ont conduit les acteurs de la normalisation vers les schémas hybrides. Les schémas actuels de compression vidéo et celui en cours d'élaboration seront ensuite détaillés pour en montrer les hautes performances, mais aussi les limites qui ont inspiré les travaux de cette thèse. Enfin les codeurs adaptés au contenu source, et orientés synthèse de texture, seront présentés.

2.1 Exploitation de la perception visuelle.

La compression de vidéos peut être composée uniquement de processus qui permettent de compresser mathématiquement le signal **sans perte** d'information. Le signal décodé est alors identique au signal source. Dans ce cas, toutes les transformations appliquées

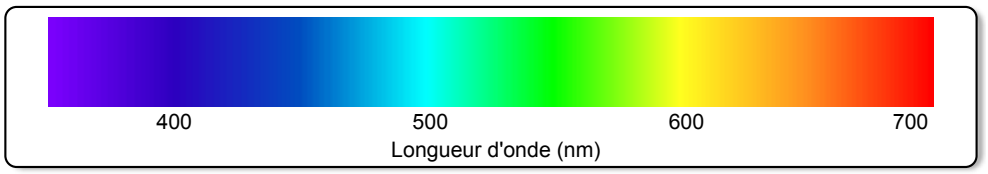


Figure 2.1 – *Spectre des couleurs.*

doivent être réversibles. Cependant, ces systèmes suffisent rarement et sont couplés aux processus de *quantification* décrits ci-après, qui tentent de réduire la quantité d'information à transmettre, tout en minimisant les dégradations des images : on parle alors de compression **avec pertes**. Dans ce dernier cas, il faut chercher à cacher, ou du moins limiter les dégradations occasionnées. Le système visuel humain (SVH) possède des propriétés de perception des couleurs, des fréquences et des contrastes, qui permettent de compresser les images en minimisant les dégradations visibles. On appelle notamment *phénomène de masquage* la diminution de la perception due à certaines interactions dans le domaine spatial et/ou temporel. À l'inverse, le phénomène de facilitation correspond aux accentuations de perception dans certaines configurations de régions d'images ou de vidéos. Les schémas de codage vont donc chercher à localiser les suppressions d'informations de préférence dans les zones de masquage afin de générer le minimum de dégradations dans les zones de facilitation.

Cette section vise à présenter quelques caractéristiques principales du SVH et leur exploitation pour la compression. La première partie propose de commencer par les propriétés du SVH face à la perception des couleurs, et ainsi décrire l'adaptation des formats d'images en couleur.

2.1.1 Représentation des couleurs et formats d'images.

Afin de pouvoir représenter toutes les couleurs d'une scène, il faut choisir un *espace* contenant toutes les couleurs du visible, afin de restituer un contenu fidèle à l'écran. Les travaux présentés dans [KGKH93] émettent la possibilité de synthétiser toutes les couleurs du spectre visible, représenté en figure 2.1, à partir de la combinaison de trois longueurs d'ondes spécifiques. Ainsi, le domaine RGB pour *Red Green Blue* permet de synthétiser les couleurs à partir des couleurs primaires rouge ($700.0nm$) vert ($546.1nm$) et bleu ($535.8nm$).

Cependant, d'après la théorie des signaux antagonistes de la perception des couleurs, le SVH distingue la lumière reçue en trois signaux de différence : {noir-blanc, rouge-vert, bleu-jaune}, issus de trois types de cônes différents, présents sur la rétine. Aussi, l'acuité visuelle chromatique est sensiblement inférieure à l'acuité achromatique. L'espace de couleurs utilisé dans le domaine de la compression, appelé YUV, ou YCbCr, est donc adapté aux caractéristiques du SVH. Y représente ainsi pour la luminance, U(Cb) correspond à la différence entre le bleu et la luminance, et V (Cr) à la différence entre le rouge et la luminance. Le passage du domaine RGB au domaine YUV se fait en appliquant la transformation suivante :

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \tag{2.1}$$

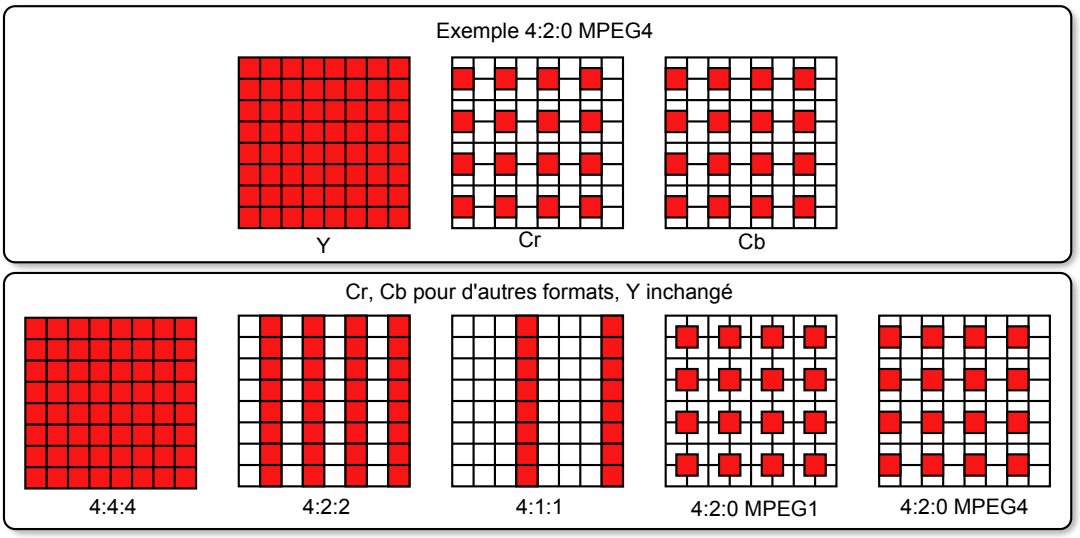


Figure 2.2 – *Formats d'échantillonnage des couleurs.*

L'acuité du SVH étant moindre pour les chrominances que pour la luminance, la toute première étape de la réduction d'information consiste généralement à choisir un format permettant la réduction des informations contenues dans les composantes chromatiques. Pour l'image numérique, cette réduction passe par un sous-échantillonnage des composantes chromatiques. La figure 2.2 illustre les formats principaux :

- $4:4:4$: pas de sous-échantillonnage.
- $4:2:2$: sous-échantillonnage horizontal d'un facteur 2.
- $4:1:1$: sous-échantillonnage horizontal d'un facteur 4.
- $4:2:0$: sous-échantillonnage horizontal et vertical d'un facteur 2. C'est le format qui est utilisé dans les principaux *codecs* (codeurs-décodeurs). On note toutefois sur la figure 2.2 que plusieurs versions de sous-échantillonnage existent.

Si chaque composante peut prendre 256 valeurs, soit un codage sur 8 bits, le mode $4:2:0$ permet par exemple de coder toute l'information de couleur pour un pixel sur 12 bits, au lieu des 24 bits nécessaires pour le mode $4:4:4$.

2.1.2 Sensibilité au contraste.

Le SVH est plus sensible aux variations locales de luminances qu'aux valeurs absolues de luminance. Pour mesurer ce phénomène, on définit la notion de *contraste* comme le rapport entre la variation locale de luminance et la luminance moyenne sur le voisinage. Mathématiquement, plusieurs définitions des contrastes sont proposées dans la littérature.

Le contraste de Weber [Cor70] exprime le rapport entre la variation locale de luminance ΔL sur un fond de luminance uniforme L dans la région considérée, soit

$$C_{Weber} = \frac{\Delta L}{L}. \tag{2.2}$$

La loi de Weber-Fechner pose que ce rapport est quasi-constant sur une large gamme de luminance et définit ainsi les bases du masquage par contraste. Une autre définition très répandue, le contraste de Michelson d'une image est défini dans [Mic95] par

$$C_{Michelson} = \frac{L_{max} - L_{min}}{L_{max} + L_{min}} \tag{2.3}$$

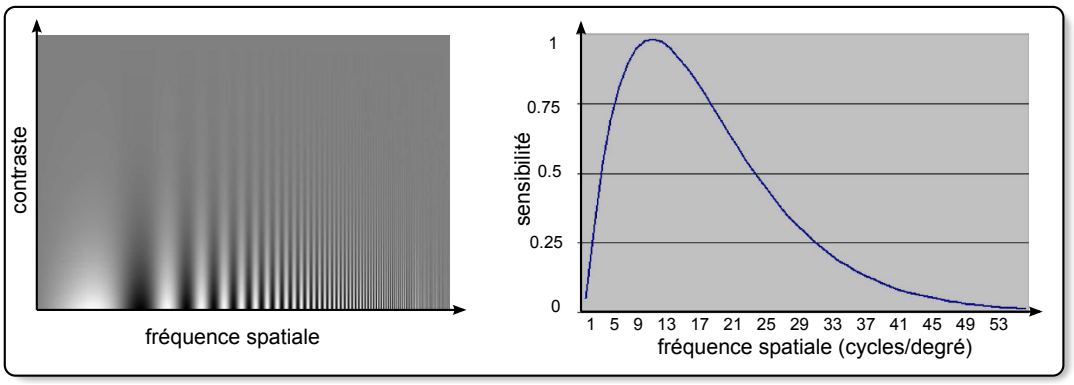


Figure 2.3 – *Sensibilité au contraste. A gauche apparaît l'illustration de la sensibilité fournie dans [CR68]. A droite est représentée l'enveloppe de visibilité normalisée proposée par J. Mannos et D. Sakrison dans [MS74].*

où L_{min} et L_{max} sont les valeurs de luminance minimum et maximum du signal. Cette définition a permis d'étendre l'étude du seuil de sensibilité au contraste du SVH, en prenant en compte d'autres caractéristiques influant sur la perception humaine. On détaillera notamment dans les paragraphes suivants les influences des fréquences spatiales et temporelles.

2.1.3 Système Visuel Humain et fréquences spatiales.

Dans le domaine de l'image, la notion de fréquence spatiale est associée à la rapidité de variation du signal dans les deux dimensions de l'image, par analogie à la fréquence temporelle. L'illustration de Campbell-Robson [CR68], en figure 2.3, présente une modulation sinusoïdale de la luminance suivant l'axe horizontal, alors que le contraste varie exponentiellement suivant l'axe vertical. On s'aperçoit alors que les bandes apparaissent plus hautes au milieu que sur les bords. Cette propriété est intéressante pour la compression d'images puisque l'œil ne distinguera pas ou peu les pertes au niveau des hautes fréquences et des faibles contrastes. La fonction de sensibilité au contraste ou CSF (Contrast Sensitivity Function) représente l'enveloppe du seuil différentiel de visibilité. Cette fonction permet de traduire mathématiquement la propriété précédente et ainsi d'adapter les processus de compression pour que les pertes soient localisées au delà de ce seuil de visibilité. On retrouve la forme de cette enveloppe sur la CSF isotrope proposée par J. Mannos et D. Sakrison dans [MS74] et présentée sur la figure 2.3 à droite.

2.1.4 Système Visuel Humain et fréquences temporelles.

Par analogie avec la sensibilité aux fréquences spatiales, la perception du SVH varie en fonction de la fréquence temporelle. Dans ses travaux publiés dans [DL+58], H. De Lange montre que la sensibilité maximale se situe aux environs de 8 Hz, et décroît rapidement pour les fréquences supérieures. De fait, si la sensibilité est acceptable pour la fréquence standard de 25 images par seconde, le rendu est quasi parfait pour les séquences cadencées à 100 images par seconde.

Après avoir détaillé quelques aspects essentiels de la vision humaine, la suite de ce chapitre présente des techniques de compression des images et des vidéos. Deux approches

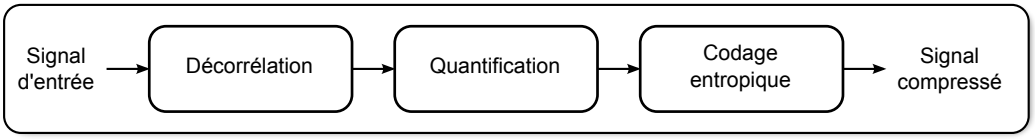


Figure 2.4 – *Structure générale d'un schéma de compression d'images.*

se complètent à cette fin : réduire les redondances qui s'y cachent, mais aussi exploiter les caractéristiques du SVH exposées précédemment pour dissimuler au mieux les distorsions.

2.2 Les principes généraux de compression d'images fixes et animées.

Les systèmes de compression d'image contiennent généralement trois étapes principales, rappelées dans la figure 2.4. La compression du signal contenu dans une image, passe d'abord par sa **décorrélation**, c'est à dire la recherche de la suppression, ou du moins la minimisation des redondances. Pour cela, deux techniques sont généralement utilisées.

- *La prédiction* : on cherche dans cette étape à prédire au mieux le signal à partir d'une autre. En effet, si deux régions identiques ou quasi identiques sont contenues dans l'image, il n'est pas nécessaire de transmettre les deux régions. Il suffira au décodeur de reconstruire la deuxième à partir de la première.
- *La transformation* : cette étape consiste à calculer la représentation du signal dans un autre domaine permettant de le traiter dans un espace adéquat.

Comme aucune des deux techniques n'offre à elle seule une décorrélation parfaite, elles sont généralement utilisées conjointement afin de minimiser l'entropie du signal à transmettre. Une dernière étape, le **codage entropique**, consiste ensuite à réduire le nombre de bits nécessaires à la représentation du train binaire envoyé. Cette étape du codage, couplée à la mise en forme du train binaire est classiquement réversible.

Dans le cas où ces outils ne suffisent pas à réduire suffisamment la quantité d'informations à transmettre, une étape de **quantification** est appliquée avant le codage entropique, introduisant des pertes d'informations. Il s'agit alors de trouver un compromis entre compression et qualité visuelle du signal décodé puis reconstruit.

Ces différents processus sont détaillés dans les sections suivantes, en commençant par évoquer les principales techniques de prédiction, fondamentales pour la compression de contenu, mais aussi, indissociable des approches utilisées pour la synthèse de texture.

2.2.1 La prédiction.

Le codage prédictif vise à approximer au mieux la valeur d'un pixel ou d'un bloc de pixels de l'image à partir du contexte *causal*. La notion de causalité se réfère d'abord au temporel : ce qui est passé par rapport à l'action en cours. Lors du codage ou du décodage d'une image fixe, cette causalité correspond à la partie spatiale connue, qui a déjà été reconstruite. Les images sont généralement traitées dans l'ordre *raster scan*, ce contexte correspond aux pixels au dessus et à gauche du bloc courant. Dans le cas du codage de séquences vidéo, le domaine causal correspond non seulement aux parties de l'image courante déjà reconstruites mais aussi aux images précédemment décodées de la séquence.

L'idée simple réside dans le fait que le décodeur puisse reproduire la prédiction au codeur, sans forcément avoir à transmettre d'informations annexes. On distingue ici deux types de prédiction possibles :

- la prédiction **Intra** qui correspond à la prédiction d'un pixel ou bloc courant uniquement à partir des informations spatiales de la partie causale de l'image courante.
- la prédiction **Inter** qui correspond à la prédiction du bloc à partir de certaines images précédemment décodées, appelées *images de référence*. Le codage des vidéos atteint des taux de compression nettement plus élevés que pour les images, du fait de la prise en compte des fortes similarités entre les images successives des séquences sources.

Les techniques relatives à ces différents modes seront détaillées à travers la présentation des standards usuels de compression. L'*erreur de prédiction* ou *résidu* est estimée simplement par la différence entre la prédiction et les valeurs réelles : c'est précisément cette information qui est encodée puis transmise au décodeur. Le résidu possède ainsi une entropie moindre, comparée à celle du signal source. La qualité de la prédiction est donc essentielle dans le schéma de compression puisqu'elle conditionne radicalement la quantité d'information résiduelle à traiter et à transmettre.

La section suivante propose de détailler les transformations principales, utilisées en compression d'images et de vidéos

2.2.2 La transformation.

La transformation est une autre technique permettant de décorréler le signal. Elle peut être utilisée seule ou appliquée sur le résidu, en sortie de la prédiction. Parmis les outils de transformation les plus usitées, on trouve :

- la **transformée de Karhunen-loève (KLT)**. L'efficacité de cette transformée réside dans le fait que ses fonctions de bases s'adaptent au signal source. En effet, le domaine transformé est constitué des vecteurs propres de la matrice de covariance, il est donc *orthogonal*. Il est de plus *optimal* puisque tous les éléments de la matrice sont décorrélés. De par la diagonalisation de sa matrice de covariance, la KLT permet de concentrer l'énergie sur les composantes principales du signal. Cependant, sa force fait aussi sa faiblesse puisqu'il faut calculer la nouvelle base et que cette étape est complexe à mettre en œuvre. En pratique elle n'est donc pas utilisée, mais elle constitue une borne supérieure d'efficacité à atteindre.
- la **transformée de Fourier** Contrairement à la KLT, la base d'arrivée est fixe et des algorithmes rapides tels que la FFT (Fast Fourier Transform) permettent d'en faire un outil plus léger à utiliser. La transformée discrète ou DFT est donnée par

$$F(k,l) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} I(m,n)\, \mathrm{e}^{-\frac{2i\Pi km}{N}}\, \mathrm{e}^{-\frac{2i\Pi ln}{N}} \tag{2.4}$$

  pour une image I ou un bloc de taille $N \times N$. Cette transformée présente cependant l'inconvénient de fournir un signal complexe.
- la **transformée en cosinus discrète (DCT)**. Outil de transformation le plus souvent utilisé dans les standards de compression d'images et de vidéos, cette DCT en 2D séparable est construite en calculant les DCT 1D sur les lignes puis les colonnes séparément. Ainsi, son calcul suit

$$F(u,v) = \frac{1}{4} C_u C_v \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} I_{mn} \cos(u\pi \frac{2m+1}{2N}) \cos(v\pi \frac{2n+1}{2N}) \tag{2.5}$$

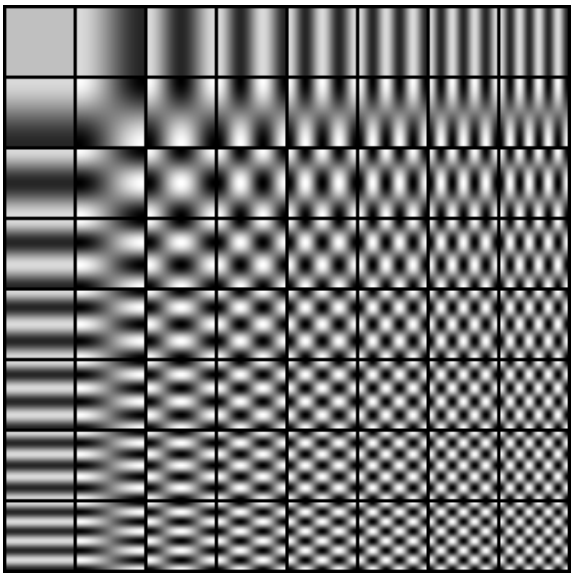


Figure 2.5 – *Fonctions de base 2D de la transformée en cosinus discrète.*

où

$$C_u = \begin{cases} \frac{1}{\sqrt{2}} & \text{si } u = 0, \\ 1 & \text{sinon,} \end{cases} \quad \text{et } C_v = \begin{cases} \frac{1}{\sqrt{2}} & \text{si } v = 0, \\ 1 & \text{sinon.} \end{cases} \tag{2.6}$$

Cette transformée est aussi orthogonale et permet une bonne distribution de la variance, ce qui implique une concentration efficace de l'énergie dans le domaine transformé. Quand le signal source répond aux critères d'un champ de Markov du premier ordre, l'efficacité de la DCT s'approche de celle de la KLT dans laquelle les fonctions de base dépendent du signal source. La figure 2.5 illustre les fonctions de base de la DCT pour un bloc de taille 8×8.

Cependant, la DCT doit être appliquée à des blocs de taille restreinte de l'image, typiquement 8×8 ou 4×4 pixels, pour limiter les distorsions. En effet, les fonctions de base de la DCT n'étant pas spatialement localisées, l'étape suivante de quantification, lorsqu'elle est très significative, introduit des artefacts visibles sous la forme de frontières rectilignes : les effets de blocs.

Une fois le signal représenté et concentré dans le domaine transformé, l'étape suivante consiste à le quantifier pour diminuer encore la quantité d'information transmise.

2.2.3 La quantification.

Cette opération consiste à associer à une valeur réelle une autre valeur appartenant à un ensemble discret. Deux cas peuvent avoir lieu : soit la source est à valeurs dans un ensemble continu soit dans un ensemble discret dont il faut réduire l'échelle. La précision des valeurs prises par le signal est alors réduite, afin que le signal quantifié puisse être décrit avec moins d'informations que sa version originale. La quantification est l'étape de la compression qui introduit donc inévitablement des *pertes* de données, rendant le schéma irréversible. Les pertes seront représentées par une erreur de quantification e_q. L'ensemble d'arrivée, appelé *dictionnaire* est discret et fini. Le cas de la quantification vers un dictionnaire de dimension 1 est appelé *quantification scalaire* : c'est celui qui est utilisé dans les codeurs de l'état de l'art et qui sera détaillé dans les paragraphes suivants.

Quantification scalaire uniforme / non uniforme

Soit un signal $X = \{x_1, .., x_N\}$ à valeurs dans l'intervalle $[a, b]$. Les valeurs de l'espace de départ $\{d_i \in [a, b] | i \in [1, N]\}$ définissent un ensemble d'intervalles $[d_i; d_{i+1}]$. A chaque intervalle $[d_i; d_{i+1}]$ correspond alors une valeur r_i unique appelée niveau de reconstruction. Dans le cas d'une quantification scalaire uniforme, $\forall i \in [1, N-1]$ $d_{i+1} - d_i = L$ est constant.

L'idée de pratiquer une quantification non uniforme parait intéressante puisqu'adaptable au contenu à coder. Il faut néanmoins, dans ce cas, transmettre un dictionnaire répertoriant les intervalles utilisés. La méthode suivante permet un compromis largement utilisé pour la compression d'images.

Quantification à zone morte

La seule modification par rapport à la quantification uniforme réside dans l'agrandissement d'un intervalle autour de $x = 0$ alors que les autres intervalles restent inchangés de valeur L. Ceci est très utile dans la compression d'image puisqu'après l'étape de transformation décrite précédemment, de très nombreuses valeurs sont condensées sur de faibles niveaux. De plus, elles correspondent souvent à des fréquences très peu perceptibles à l'œil humain, qui pénalisent le système de codage alors qu'assignées à 0, elles ne sont pas prises en compte par le codeur entropique, dernière étape du processus d'encodage.

Après avoir passé les étapes de décorrélation par prédiction et transformation, puis la quantification, un algorithme de codage entropique permet finalement, en utilisant la théorie de l'information, de réduire la quantité de bits nécessaires à la description du signal transmis.

2.2.4 Codage entropique réversible.

Le codage mathématique vise à changer la représentation du signal, afin de minimiser le nombre de bits nécessaires pour le décrire. Comme il a été évoqué en introduction, pour mesurer cette quantité minimum, on définit l'entropie de Schannon H. On note les éléments $x_i, \forall i \in [1, N]$ d'un signal X, représentés par les symboles s_i, codés chacun sur un nombre de bits variable. Théoriquement, $H(X)$ correspond à la borne inférieure de la quantité de bits moyenne par symbole pour coder X sans perte d'information. L'entropie correspondant à l'incertitude du récepteur par rapport au signal transmis : cette propriété est vraie si le décodeur ne possède pas d'informations a priori sur les symboles reçus. Il est donc possible de descendre sous cette barre si le décodeur possède des informations contextuelles sur le signal transmis.

Formellement, si $P(X = x_i)$ est la loi de probabilité des évènements de X, on a :

$$H(X) = -\mathbb{E}(\log_2 P(X = x_i)) = -\sum_{i=1}^{N} P_i \log_2(P_i). \tag{2.7}$$

Un *codeur entropique* vise donc à construire des symboles s_i d'une longueur moyenne proche de $H(X)$, à partir des caractéristiques du signal d'entrée X. Les symboles n'ayant pas tous la même longueur, les plus courts sont affectés aux évènements qui apparaissent le plus fréquemment. C'est le principe des codes à longueur variable (VLC pour Variable Length Code). Parmi les codeurs entropique les plus célèbres, on peut citer le codage de Huffman, le codage arithmétique ou encore CABAC (Codage Arithmétique Binaire à Contexte Adaptatif) utilisé dans le standard H.264/AVC.

Norme	débits classiques	Applications
H.261	64kbit/s	Visioconférence (ISDN)
MPEG-1	1.5kbit/s	Vidéo à la demande (VOD) CD-ROM Visioconférence
MPEG-2, H.262	1.5kbit/s 1.5 - 9.72 Mbit/s 10- 20 Mbit/s	CD-ROM DVD TVHD
H.263	64Kpit/s 1.5 Mbit/s	Visioconférence (ISDN) Visioconférence (WAN)
MPEG-4/AVC, H.264	64 Kbit/s 56Kbit/s - 1Mbit/s 1Mbit/s 6Mbit/s	Visioconférence (ISDN) VOD CD-ROM TVHD

Table 2.1 – *Débits et applications visées des principales normes de compression vidéo.*

2.3 Les techniques développées dans les standards de compression.

Cette section présente les différents algorithmes et techniques mis en œuvre dans les standards les plus utilisés du domaine de la compression de vidéos. Ces méthodes sont dites hybrides du fait qu'elles combinent codage prédictif et codage par transformée. Après un bref historique, les principales caractéristiques communes aux standards seront présentées. Enfin, seront détaillées les spécificités du standard de référence actuel (H.264/AVC) ainsi que son successeur en cours d'élaboration (HEVC).

2.3.1 Historique des normes et standards

Les premiers besoins en compression de vidéo numérique proviennent du domaine de la visioconférence. La première norme en ce sens, H.261, a été établie en 1990 par l'ITU (International Telecommunication Union). Ce sont cependant les normes MPEG-1 puis MPEG-2, développées par le groupe *Motion Picture Experts Group*, qui ont permis, en augmentant les débits possibles, d'accéder à des services tels que le stockage, la lecture et la diffusion de la vidéo à la demande. Enfin, la norme H.264 MPEG4/AVC, partiellement détaillée en section 2.4, qui s'était initialement spécialisée dans le codage à bas débit, est actuellement la norme utilisée pour la diffusion de la télévision en haute définition. Elle s'est imposée en apportant un gain de 50% par rapport au standard MPEG2. Le tableau 2.1 référence les principales normes avec les débits visés ainsi que les applications principales auxquelles elles sont dédiées.

Toutes ces normes ont en commun d'être définies à partir de la structure du décodeur ainsi que de la syntaxe du train binaire envoyé par le codeur. La section suivante présente la structure de cette syntaxe.

2.3.2 Syntaxe hiérarchique.

Une séquence vidéo est décrite par les standards MPEG à différents grains. La figure 2.6, présentée dans [BD96], illustre l'imbrication des niveaux suivants :

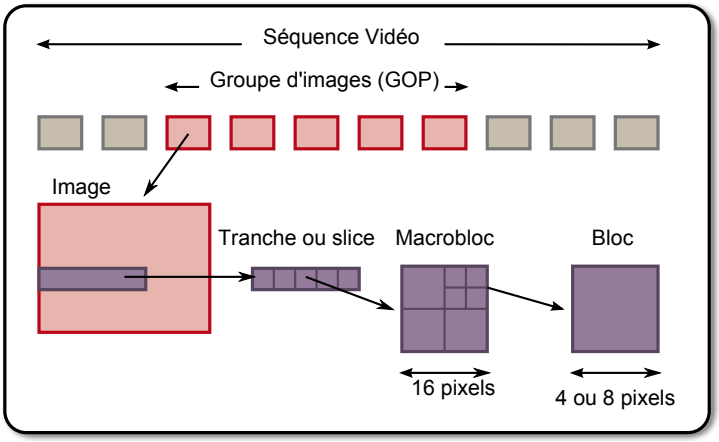


Figure 2.6 – *Les différents niveaux d'une séquence dans la syntaxe MPEG.*

- La *séquence* qui comprend les paramètres globaux tels que le nombre d'images par seconde, la taille des images, le format de représentation des couleurs.
- Le *GOP* ou groupe d'images : c'est le schéma de codage minimal qui est répété périodiquement dans le temps. Il définit ainsi la période de codage de la séquence.
- L'*Image* est l'unité de la séquence sur l'axe temporel.
- Le *Slice* définit un groupe de macroblocs (voir ci-après). C'est la partie élémentaire permettant la synchronisation du flux. Lorsqu'une partie du flux transmis est erronée, le décodeur passe au traitement du slice suivant.
- Le *Macrobloc* (MB) est un bloc de taille 16×16 pour la composante des luminances. La taille des blocs associés de chrominances dépend du format de couleur choisi : pour un format $4:2:0$ par exemple, la taille des blocs de chrominances sera 8×8.
- Le *Bloc* de taille 4×4 ou 8×8 généralement. C'est à ce niveau que s'opèrent les choix des modes de prédiction et de transformation quand le macrobloc est trop large. La taille du bloc élémentaire permet ainsi de s'adapter à l'activité locale du signal source.

Les parties suivantes détaillent les techniques et structures supplémentaires permettant de prendre en compte la séquence à tous les niveaux précédemment décrits. Les principales contributions par rapport au codage d'images fixes résident principalement au niveau des étapes de prédiction temporelle. Au niveau du GOP d'abord, le fait de pouvoir prédire des données spatio-temporelles nécessite de définir une structure particulière.

2.3.3 Les différents types d'images pour la prédiction.

A l'intérieur d'un GOP, on distingue trois types d'image :

- *Les images I* pour lesquelles tous les macroblocs sont codés en mode Intra : elles sont indépendantes des images voisines et constituent donc les images de références. Ce sont celles qui consomment le plus de ressources binaires. Elles permettent cependant de se resynchroniser sur le signal source.
- *Les images P* sont des images prédites temporellement à partir d'une image de référence dans le passé. Les macroblocs sont codés soit en Inter, soit en Intra. Ces images sont codées par différence avec une image passée de type I ou P.
- *Les images B* sont des images bi-prédites temporellement à partir de deux images de référence situées, généralement, respectivement dans le passé et dans le futur. Les

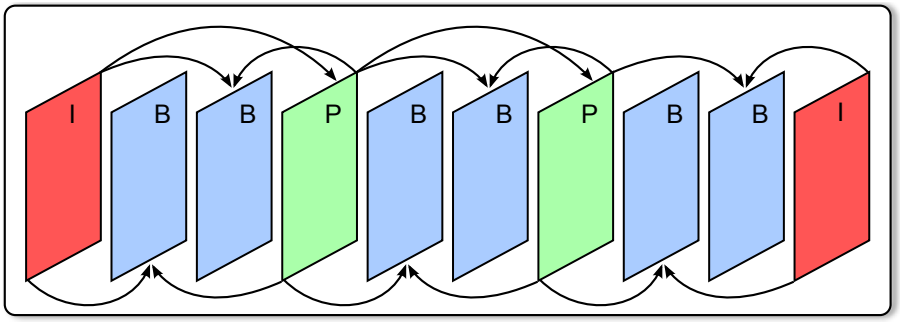


Figure 2.7 – *Exemple d'enchaînement des types d'images pour la prédiction* Inter.

macroblocs peuvent là encore être codés soit en Inter, soit en Intra. L'intérêt de ce type d'images réside dans leur moindre coût par rapport aux autres et permettent d'introduire de la scalabilité temporelle, *i.e.* la possibilité de décoder la vidéo à différentes cadences. Cependant, elles nécessitent une gestion particulière des images puisque l'ordre de décodage des images n'est plus celui de l'affichage.

Un exemple d'enchaînement de ces images dans un GOP est illustré en figure 2.7. Ainsi, le codage des images I ne contient qu'une étape de prédiction intra. Pour les autres, les modes de codage intra et inter seront en compétition.

2.4 Le standard H.264 MPEG-4/AVC.

Le standard H.264/MPEG-4 AVC, né en 2003, est issu du projet H.26L [SW02] des travaux de l'équipe JVT (Joint Video Team). Cette dernière est constituée des groupes ITU-T / VCEG et de ISO / MPEG. Il apporte une amélioration des performances de compression de l'ordre de 50% par rapport à H.262 / MPEG-2, le standard précédent.

La norme MPEG4-partie 2 a d'abord permis d'introduire la notion d'objets audio et vidéo. Elle comporte deux grandes parties : un ensemble d'outils de codage pour l'audio et la vidéo, et un langage syntaxique pour décrire les objets audio / vidéo et les outils. La partie objet prévoit notamment une description des objets par maillage, et un codage spécifique pour chaque type de données à coder (mouvement, texture, et forme des objets). Par la suite, MPEG4 partie 10, intitulé H.264/MPEG-4 AVC sera principalement utilisé pour ses performances de codage. Dans la suite de ce manuscrit, H.264/MPEG-4 AVC sera noté H.264 pour simplification.

Comme pour les standards précédents, la norme H.264 définit uniquement la syntaxe du flux binaire et la structure du décodeur. Alors que le décodeur ainsi défini doit être capable d'interpréter le train binaire reçu et que la syntaxe du flux répond à la norme, le codeur peut lui être modifié pour son optimisation ou l'adaptation à un contexte de transmission de vidéos. Le codeur H.264 est illustré sur la figure 2.8. Comme pour les normes précédentes, le codeur hybride exploite les deux décorrélations : prédiction et transformation. La boucle illustrée en les prédictions intra et inter, ainsi que les spécificités de transformation et de codage entropique.

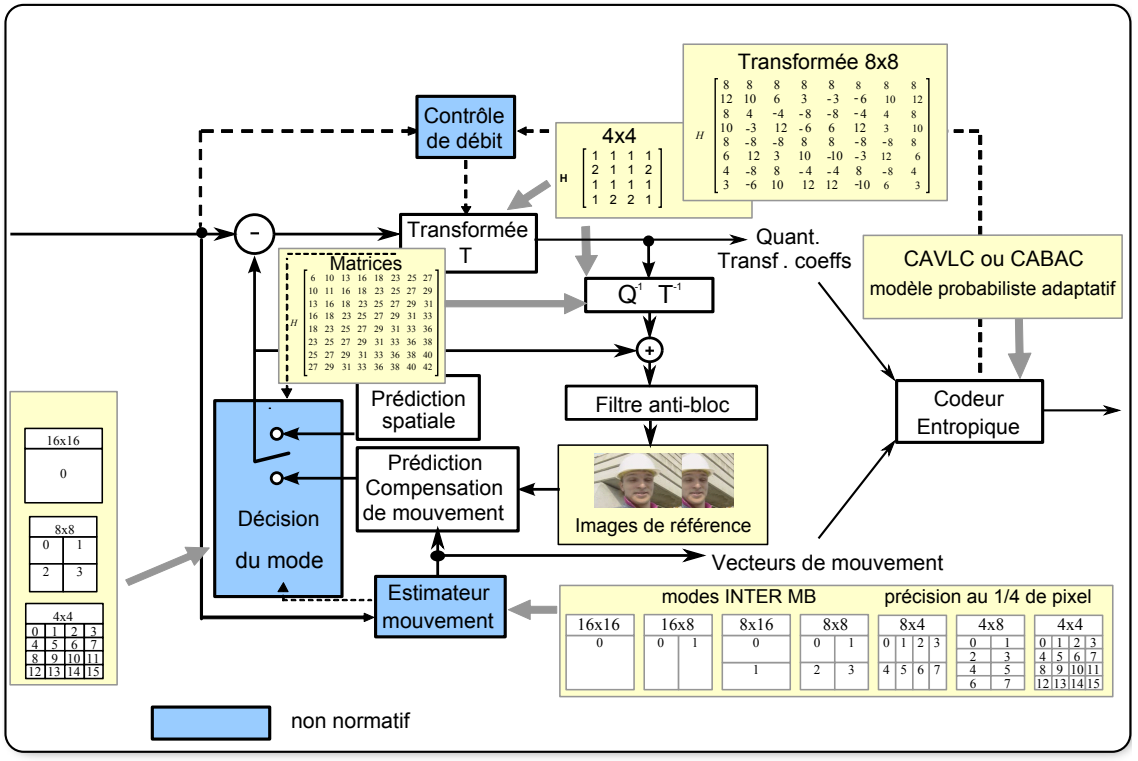


Figure 2.8 – *Spécificités d'un codeur MPEG4/AVC.*

2.4.1 Prédiction intra-images.

Contrairement aux standards précédents, pour lesquels la prédiction intra s'opère dans le domaine transformé, les modes H.264 sont calculés dans le domaine pixel. Le macrobloc à coder est donc prédit (c'est à dire approximé) à partir de ses voisins spatiaux (choisis parmi le voisin gauche et les trois voisins supérieurs). Pour cette étape, le standard, dans sa version finale considère trois tailles de bloc : 16×16, 8×8 mais aussi 4×4, offrant ainsi une granularité accrue par rapport à la précision des standards précédents.

Prenons l'exemple d'un bloc qui sera déduit des pixels du dessus qui ont déjà été décodés, ce qui correspond au mode vertical (figure 2.9). L'opération consiste simplement à répéter 4 fois la dernière ligne du bloc supérieur, de manière à créer un nouveau bloc 4×4. Ce nouveau bloc est appelé prédiction spatiale et permet d'approximer le bloc à coder. L'erreur commise (différence des valeurs pixel à pixel entre le bloc source et sa prédiction), devra être transmise au décodeur. Ce schéma est valide pour les blocs 4×4 et 8×8 de luminance. Pour les blocs 16×16 de luminance et 8×8 de chrominance, 4 modes sont disponibles : horizontal, vertical, DC et un mode nommé *plane* correspondant à un plan linéaire construit à partir des pixels de bord causaux.

2.4.2 Prédiction inter-images

Cette section décrit le principe de prédiction inter-images qui permet au codeur de compresser les séquences d'images avec une efficacité bien supérieure à celle d'un codeur d'images fixes, appliqué image par image. Après la brève présentation des outils utilisés dans les codeurs MPEG, cette section vise à détailler les améliorations apportées dans ce

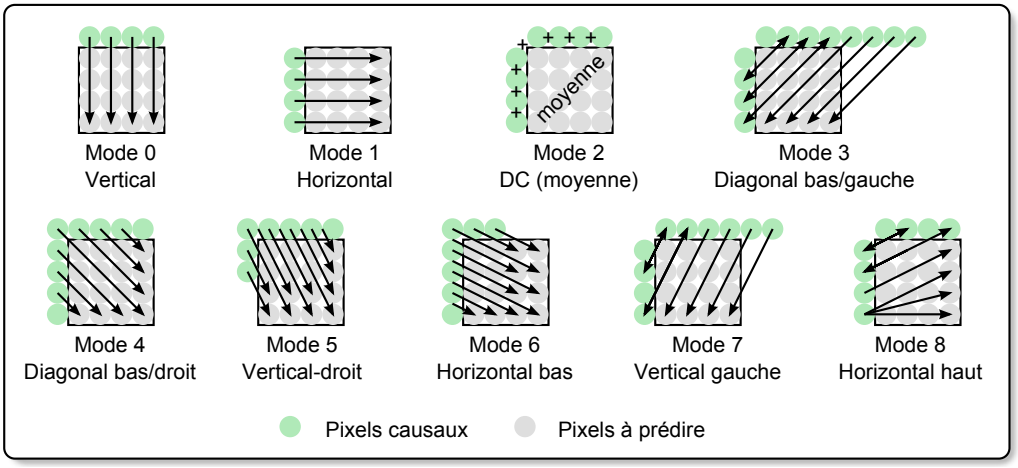


Figure 2.9 – *Les modes* intra *pour la prédiction d'un bloc* 4×4.

domaine par le standard H.264.

Estimation et compensation de mouvement.

Les outils d'estimation [JJ81] et de compensation de mouvement ont considérablement réduit les débits nécessaires à la transmission de séquences vidéo. Certaines applications spécifiques aux codeurs hybrides de la compensation de mouvement sont notamment détaillées dans [Gir87]. Dans le cadre du codage inter, le codeur est doté d'un estimateur de mouvement. Celui-ci a pour rôle de déterminer le déplacement de la sous-partition courante (bloc, macrobloc) vis-à-vis d'une image précédemment codée/décodée (appelée image de référence). En règle générale, les algorithmes d'estimation de mouvement sont de type *block matching*. Cette technique, illustrée par la figure 2.10, permet de trouver dans une image de référence le bloc maximisant une mesure de corrélation avec le bloc courant. Le bloc de l'image de référence trouvé par l'estimateur est appelé prédiction temporelle.

Classiquement, à chaque bloc est associé un vecteur de mouvement. Ce vecteur est ensuite codé et transmis au décodeur. Comme pour la prédiction spatiale, il est ensuite nécessaire de transmettre l'erreur de prédiction. Les partitions possibles du macrobloc pour cette opération sont illustrées sur la figure 2.11. On note qu'une première partition s'opère au niveau du macrobloc 16×16. Puis, si une partition 8×8 est choisie, les blocs 8×8 peuvent encore être partitionnés de la même manière.

Dans cette quête de minimisation des redondances temporelles, on peut citer les contributions suivantes, dans le cadre de la prise en compte du mouvement par rapport au précédent standard MPEG-2.

- Le découpage des macroblocs possibles jusqu'à 4×4 est illustré par la figure 2.11, par rapport au MB classique de 16×16 pixels.
- La compensation de mouvement se fait avec une précision au quart de pixel.
- La **bi-prédiction** :

Une des spécificités du standard H.264 est de permettre la prédiction d'images B, présentées précédemment, à partir de plusieurs images antérieures et futures dont d'autres images B précédemment codées/décodées. Ces images B pouvant servir de références sont appelées *B-stored*. La figure 2.12 présente un tel enchaînement avec des images B ayant 3 images de référence possibles.

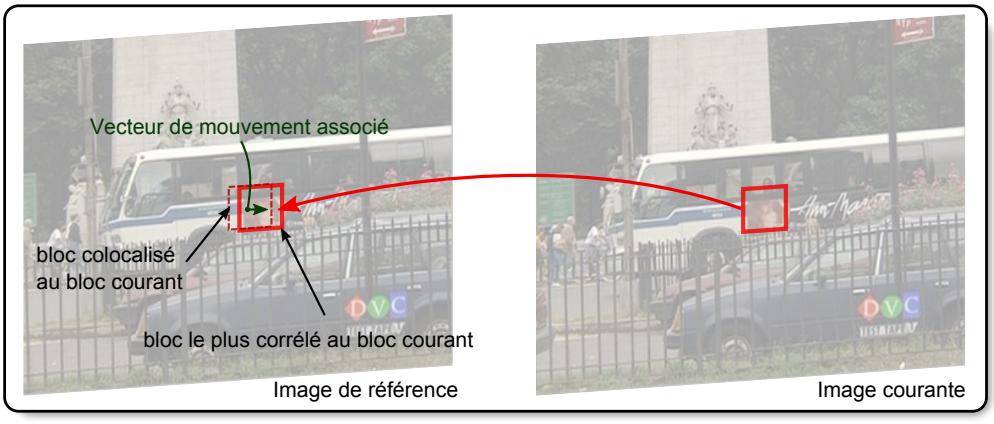


Figure 2.10 – Block matching *: recherche de corrélation dans une image de référence.*

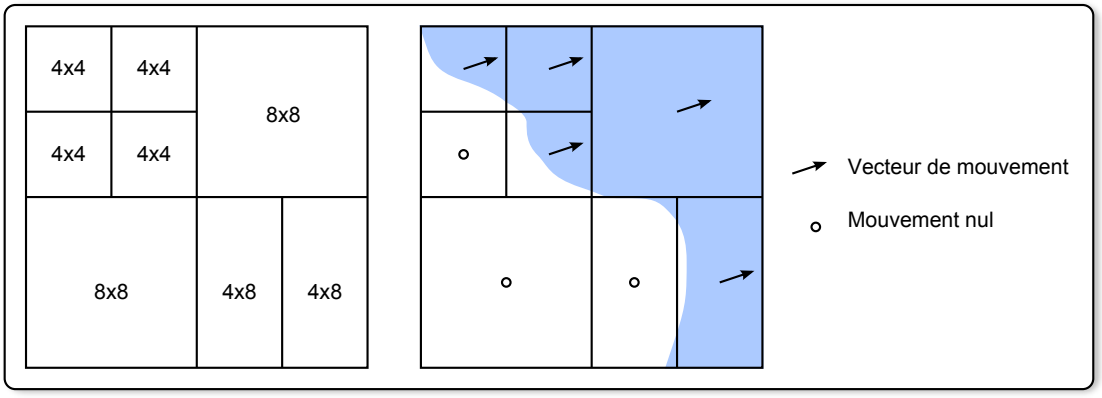


Figure 2.11 – *Exemple de partition d'un macrobloc pour la prédiction inter.*

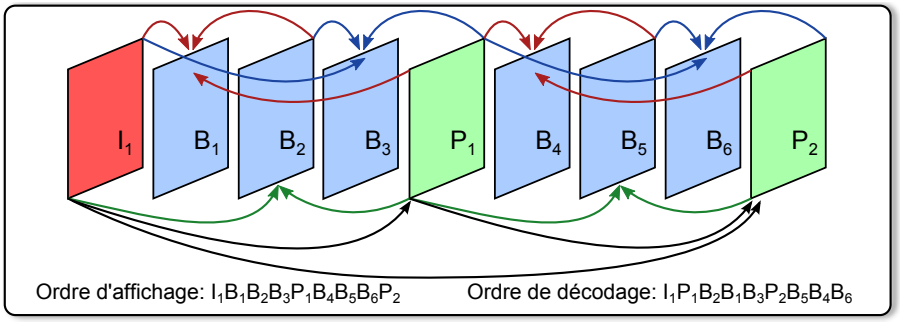


Figure 2.12 – *Structure de GOP comportant des images B bi-prédites, possibilité nouvelle du standard H264.*

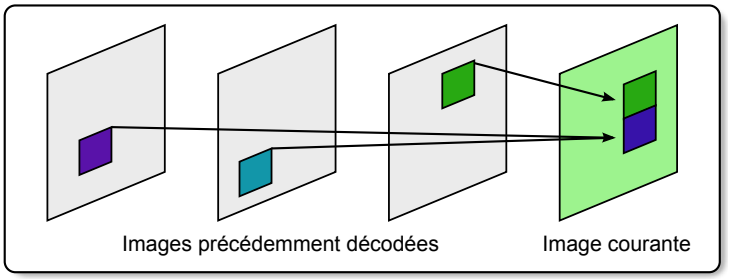


Figure 2.13 – *Références multiples pour la prédiction inter d'un bloc.*

Le groupe d'images présenté en figure 2.12 possède une structure appelée *B hiérarchique* puisque l'image B_2 est d'abord décodée afin de servir de référence pour les images B_1 et B_3. Cette structure permet, au dernier étage, une prédiction avec des images directement voisines, ce qui améliore la qualité de corrélation lors des mouvements rapides. Le fait de mettre en œuvre cette structure permet de gagner jusqu'à 15% de débit à qualité égale sur certaines séquences, comparé à H.264 sans bi-prédiction hiérarchique.

Par ailleurs, pour des images P et B, il est possible d'utiliser plusieurs macroblocs provenant de plusieurs images de références, afin de prédire le macrobloc courant. Pour les images P, le processus reste monodirectionnel puisqu'un seul prédicteur est choisi, mais il peut maintenant être choisi dans plusieurs images de référence. Pour les images B, deux macroblocs sont utilisés, et il est possible de changer d'image de référence pendant le processus de prédiction de l'image en cours. La figure 2.13 montre ainsi que deux macroblocs d'une même image peuvent être prédits à partir d'images de référence différentes : la flexibilité de la prédiction est ainsi accrue.

Après avoir détaillé les deux grandes catégories de prédictions, la section suivante présente comment le codeur peut choisir le mode de codage approprié au MB en cours. Les normes étant écrites à partir du décodeur, ces méthodes ne sont pas directement liées à un standard, et en particulier à H.264.

2.4.3 Choix des modes de codage.

La syntaxe du flux envoyé au décodeur comportant les informations relatives aux modes choisis, le codeur est libre pour la sélection des modes. Ce choix est cependant conditionné par le type d'image traitée : seule la prédiction intra sera par exemple active dans le cas

d'une image I. L'optimisation débit/distorsion, ou RDO (Rate/Distorsion Optimization), est le processus le plus souvent utilisé pour décider du meilleur mode parmi les modes et sous-modes intra et inter. Le but de cette étape détaillée dans [SW98, OR98] est alors de minimiser la distorsion sous la contrainte d'un certain débit. Selon [WSJ+03], cette opération consiste trouver pour un signal S et un mode de prédiction I, selon

$$\min_I D(S.I) \text{ soumis à } R(S,I) \leq R_c \tag{2.8}$$

avec $D(S.I)$ le critère de distorsion, $R(S.I)$ le débit et R_c la contrainte de débit. En pratique, cette opération revient à minimiser une fonction Lagrangienne donnée par

$$J_{lambda}(S,I) = D(S,I) + \lambda.R(S,I), \tag{2.9}$$

avec $\lambda \geq 0$ un paramètre permettant de pondérer la fonction entre ces deux critères. Le processus a donc deux options pour faire ce choix en tenant compte des deux paramètres :

- Méthode *a posteriori* : dans ce cas, les étapes suivantes de l'encodage d'un macrobloc *i.e.* la transformée spatiale, la quantification et le codage entropique, sont nécessaires pour connaître le débit. De plus, le calcul de la distorsion par rapport au signal source, nécessite d'appliquer les opérations inverses de décodage. De manière optimale, cette étape est appliquée pour tous les modes afin de trouver celui qui minimise la distorsion tout en atteignant la contrainte de débit simultanément. Cependant, ces opérations étant coûteuses, notamment les transformées, les codeurs utilisant la RDO *a posteriori* ne testent que les modes les plus probables.
- Méthode *a priori* : dans ce cas, un modèle est utilisé pour estimer le débit nécessaire et la distorsion correspondante. Ce modèle est empirique et permet d'approcher les performances des tests a posteriori. Cette méthode sous-optimale réduit cependant nettement la complexité du choix de mode de codage.

Le paragraphe suivant décrit succinctement le type de transformée utilisée dans le standard H.264.

2.4.4 Transformées discrètes.

Afin de coder les résidus, une transformée DCT *entière* est appliquée sur les blocs 4×4 et 8×8 contrairement aux précédentes normes où seule la DCT 8×8 était utilisée. Cette évolution permet de maximiser la cohérence avec la prédiction, qui peut aussi s'appliquer sur des blocs de 4×4 ou 8×4 et 8×4. Les fonctions de base de la DCT n'étant pas spatialement localisées, la différence de quantification d'un bloc à l'autre introduit une discontinuité entre les blocs adjacents. C'est pourquoi un filtre contre les effets de bloc (deblocking filter) a été introduit dans le décodeur défini par la norme H.264.

2.4.5 Nouveau codeur entropique.

La norme H.264 comprend deux codeurs entropiques : le CAVLC pour Context Adaptive Variable Length Coding et le CABAC pour Context Adaptive Binary Arithmetic Coding. Le but clairement affiché réside dans l'adaptation maximum au contenu à encoder. Les caractéristiques du codeur ne sont pas directement liées aux travaux de cette thèse qui trouvent plus d'échos dans les étapes de prédiction.

2.4.6 Vers le standard HEVC.

Le standard HEVC pour High Efficiency Video Coding [HEV11] est le futur successeur de H.264, en cours de développement par l'équipe Joint Collabortive Team on Video Coding (JCT-VC) [HEV10a], commune à MPEG et VCEG. Comme son nom l'indique, il vise à améliorer les performances de codage de son prédécesseur, c'est à dire diviser par deux la taille du train binaire nécessaire au décodage d'images de qualité comparable à H.264 avec ses paramètres les plus performants [HEV10b]. Ainsi, HEVC permettra de diffuser la télévision numérique pour les futures générations d'écrans en ultra haute définition, soient 7680×4320 pixels.

Le schéma est toujours en développement et des modifications sont approuvées au fur et à mesure, on peut néanmoins citer quelques outils à l'origine de cet accroissement de performance. Telles qu'elles sont présentées dans le modèle de test d'octobre 2010, les principales améliorations sont portées à tous les niveaux :

- les tailles de macroblocs considérés pour le codage, allant maintenant de 8×8 à 64×64 ;
- la transformée s'opère sur des blocs de tailles 4×4 à 32×32 ;
- le nombre de directions possibles pour les modes intra passe à 34 ;
- un choix adaptatif des matrices de quantification utilisées ;
- de nouveaux filtres pour limiter les effets de bloc...

Ces contributions doivent cependant être mises en œuvre avec le souci de conserver une complexité acceptable. Le calendrier prévoit une version finale, prête à être standardisée en janvier 2013. Comme pour ses prédécesseurs, il restera ensuite au standard à s'imposer comme l'outil incontournable de transmission et stockage de contenus vidéo.

2.4.7 Conclusion sur les standards.

L'évolution des techniques utilisées, couplant avancées technologique et algorithmiques, apportent une grande efficacité de codage pour les applications d'aujourd'hui, avec le standard H.264. Cependant, les écrans et les contenus média continuent d'évoluer, ce qui nécessitera toujours des performances de codage accrues, ou du moins des outils de codage dédiés à ces applications. C'est pourquoi l'équipe JCT-VC développe HEVC, et devra développer de nouveaux outils. Les travaux présentés dans cette thèse, ainsi que les schémas du même type de la littérature, ne sont pas développés pour faire l'objet de propositions pour le standard HEVC, puisqu'éloignés de l'optique de choix de mode sur critère débit/distorsion objectifs, et vraisemblablement pas assez matures pour un standard à court terme. Il contribue donc aux recherches en amont qui pourront permettre la construction de futurs standards en rupture, dont les applications nécessiteront encore plus de réductions de débits. Après avoir détaillé l'évolution des codeurs de vidéos, la section suivante présente comment évaluer leurs performances et quels sont les enjeux de ces critères pour les futurs schémas.

2.5 La compression et les mesures de distorsion.

La mesure de distorsion se divise en deux grandes catégories.

- La mesure **objective** qui se fonde sur une estimation de la distorsion entre les pixels de l'image dégradée et ceux de l'image source. La note peut donc être donnée automatiquement par un calcul allant de la simple différence aux algorithmes prenant en compte les propriétés du SVH.

- La mesure **subjective** qui donne une note à l'image dégradée sur sa qualité perçue par un individu ou un groupe d'individus.

La première est évidemment souhaitable afin de construire un schéma automatique attestant la qualité des images décodées. Le but ultime étant d'approcher au mieux la note donnée par une mesure subjective sur un large échantillon de population. Il s'avère cependant que dans la plupart des cas, les critères objectifs ne suffisent pas à évaluer parfaitement la qualité perçue. Avant de détailler les principales méthodes d'évaluation de la qualité, la section suivante présente les distorsions qui sont occasionnées par les codecs standards de compression.

2.5.1 Les distorsions dues à la compression.

La plupart des codeurs présentés ci-dessus, malgré les multiples optimisations, reposent sur des outils similaires et produisent donc des artefacts communs. En effet, la chaîne complète (compensation de mouvement, transformée DCT, quantification des coefficients) est comparable et appliquée notamment sur des blocs dans la plupart des codeurs. Pour rappel, à l'issue de cette chaîne, seule l'étape de quantification introduit des pertes et donc des distorsions. Cependant, les artefacts produits dépendent de la chaîne entière puisque la quantification est opérée sur le résidu transformé du signal prédit. On relève plusieurs types de distorsions sur les vidéos décodées.

- *Les effets de bloc* : ils correspondent à l'apparition de frontières rectilignes visibles entre les blocs qui ont été reconstruits après décodage.
- *Le flou* correspond logiquement à la suppression des coefficients hautes fréquences lors de la quantification dans le domaine DCT. Cette distorsion est sans doute l'une des plus sensible et gênante.
- *Les saccades* peuvent apparaître lors d'une mauvaise compensation de mouvement.
- *Le color blending* : une frontière nette sépare deux régions dont les chrominances sont fortement différentes. Les hautes fréquences supprimées par l'échantillonnage $4:2:0$ par exemple, introduisent une bavure de couleurs sur le macrobloc.
- *Le papillotement (ou flickering)* apparaît principalement dans les zones texturées. La texture des blocs de ces zones est compressée avec des pas de quantification variant au cours du temps, ce qui a pour effet de créer des papillotements dans ces zones. Cette distorsion, présente sur les textures, va être déterminante dans la réussite des algorithmes de compression orientés synthèse.

Cette liste n'est pas exhaustive mais permet d'aborder les principaux artefacts qu'il va falloir appréhender.

2.5.2 Les méthodes d'évaluation subjective.

Le test subjectif permet d'évaluer la qualité d'images ou de vidéos par les notes données par un panel représentatif d'observateurs. Ainsi, ils permettent souvent de calibrer par l'expérience ce que les critères objectifs doivent approcher. Cependant, même si l'évaluation subjective permet de s'affranchir de tous les problèmes de modélisation par la vision artificielle, il existe tout de même des biais dus aux conditions de test. Ces dernières ont donc fait l'objet d'une normalisation par l'ITU [IR93], qui détermine l'ensemble des règles auxquelles doivent répondre les tests. Ces règles définissent les deux principaux facteurs suivants.

- L'environnement de visualisation qui comprend la distance entre l'écran et l'observateur, la luminosité ambiante, et l'isolement de l'écran dans le champ de vision (l'écran se trouve devant un font uniforme).

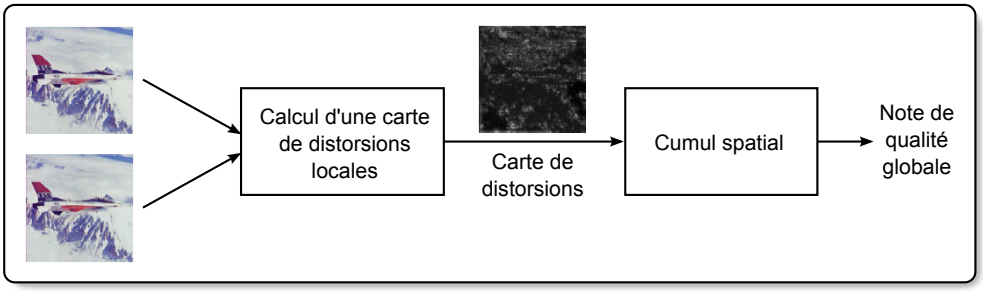


Figure 2.14 – *Structure typique d'une métrique de qualité d'image*

- Le panel d'observateurs qui doit prendre en compte les proportions de certaines caractéristiques comme l'âge, le sexe, le port de lunettes ou encore la profession.

Les données résultant des tests subjectifs doivent ensuite être traitées et interprétées. On présente d'abord la note MOS (Mean Opinion Score) qui correspond à la moyenne des notes fournies par les observateurs. Il est à noter qu'il est souhaitable de détecter les résultats incohérent d'un individu isolé pour exclure ses notes du calcul du MOS.

Les schémas développés au cours de ces travaux de thèse ont fait l'objet de tests subjectifs. Le détail des conditions de tests, relatives aux expérimentations menées, sera donné dans les chapitres 4 et 5

2.5.3 Les méthodes d'évaluation objective.

Les différentes approches détaillées ici visent, le plus souvent, à évaluer objectivement la qualité de l'image entière. Elles relèvent donc l'ensemble des distorsions présentes sur la surface de l'image pour calculer une note globale de qualité. Elles peuvent néanmoins permettre de comparer localement les distorsions produites par les modes de compression par exemple, afin de choisir le meilleur selon le critère utilisé. La figure 2.14 présentée dans [Nin09] illustre cette structure passant par la création d'une carte des distorsions avant de les cumuler pour obtenir une note globale.

Les méthodes purement signal.

Le rapport signal à bruit est le plus couramment utilisé pour évaluer la distorsion entre le signal décodé et le signal source. Le critère pour mesurer ce rapport est appelée PSNR pour Peak Signal to Noise Ratio. Son calcul suit la formule :

$$PSNR = 10.\log_{10}\left(\frac{d^2}{EQM}\right), \tag{2.10}$$

avec d correspondant à la dynamique du signal, soit typiquement $d = 255$ pour la composante d'un pixel codée sur 8 bits, et EQM signifie Erreur Quadratique Moyenne, connue aussi sous l'acronyme MSE pour Mean Squared Error. Elle est définie par :

$$EQM = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\|I_s(i,j) - I_d(i,j)\|^2 \tag{2.11}$$

entre une image source I_s et une image dégradée I_d de taille $m \times n$.

Ce calcul constitue la manière la plus simple de juger de la dégradation, pixel à pixel d'un signal par rapport à sa version d'origine. Il n'est cependant pas forcément adapté au SVH. En effet, toutes les considérations énoncées en section 2.1, comme la sensibilité au contraste, par exemple, ne sont plus prises en compte. Malgré cela, cette métrique constitue toujours la mesure de référence pour juger de l'efficacité d'un codec, ce qui pénalise les nouveaux types de codeurs orientés synthèse, décrits en section 2.7. C'est pourquoi de nouvelles métriques ont vu le jour afin de tenir compte de certains de ces aspects, notamment la structure de l'image.

Les méthode structurelles.

L'approche **SSIM** pour *Structural SIMilarity* a été développée par Wang et al. dans [WBSS04]. Fondée sur les précédents travaux présentés dans [LWBK02, WSB02], elle ne cherche pas comme le PSNR à évaluer les différences pixel à pixel entre l'image source et l'image dégradée, mais de mesurer la *similarité de structure*. L'objet de cette métrique est donc de tenter de construire une métrique objective qui tient compte des propriétés du SVH lui même très sensible aux distorsions affectant les structures de l'image. On entend par structure l'agencement des valeurs des pixels, particulièrement l'agencement local. Fort de ce constat, le calcul de cette métrique s'effectue à travers la comparaison de blocs ou de fenêtres glissantes sur les images. Soient f_x et f_y les signaux x et y regardés à travers la fenêtre f. La SSIM se compose de trois notes : la première, noté l sur les luminances, la deuxième c sur le contraste et la dernière s sur la structure. La SSIM est ainsi définie par :

$$SSIM(f_x, f_y) = [l(f_x, f_y)]^{\alpha} . [c(f_x, f_y)]^{\beta} . [s(f_x, f_y)]^{\gamma} \tag{2.12}$$

avec

$$\begin{cases} l(f_x, f_y) = \frac{2\mu_{f_x}\mu_{f_y}+C_1}{\mu_{f_x}^2+\mu_{f_y}^2+C_1} \\ c(f_x, f_y) = \frac{2\sigma_{f_x}\sigma_{f_y}+C_2}{\sigma_{f_x}^2\sigma_{f_y}^2+C_2} \\ s(f_x, f_y) = \frac{2\sigma_{f_x f_y}+C_3}{\sigma_{f_x}\sigma_{f_y}+C_3} \end{cases} \tag{2.13}$$

où μ représente la moyenne du signal sur la fenêtre f et σ la variance. Les constances C_1, C_2 et C_3 permettent de stabiliser les divisions quand le reste du dénominateur est très faible. Dans l'équation 2.12, $\alpha > 0$, $\beta > 0$, $\gamma > 0$ qui sont des paramètres permettant d'ajuster l'importance de chaque terme. Les auteurs simplifient l'expression en assignant $\alpha = \beta = \gamma = 1$ et $C_3 = \frac{C_2}{2}$. L'expression simplifiée de la SSIM est alors donnée par :

$$SSIM(f_x, f_y) = \frac{(2\mu_{f_x}\mu_{f_y} + C_1)(2\sigma_{f_x f_y} + C_2)}{(\mu_{f_x}^2 + \mu_{f_y}^2 + C_1)(\sigma_{f_x}^2 + \sigma_{f_y}^2 + C_2)} \tag{2.14}$$

avec :

- $\sigma_{f_x f_y}$ la covariance entre les deux signaux sur la fenêtre f,
- $C_1 = (k_1 L)^2$ et $C_2 = (k_2 L)^2$,
- L la dynamique du signal, typiquement 255 pour une composante codée sur 8 bits,
- par défaut, $k_1 = 0.01$ et $k_2 = 0.03$.

En appliquant la SSIM sur une fenêtre glissante centrée sur chaque pixel des images à comparer, il est possible de créer une carte des erreurs structurelles comme illustré sur la figure 2.15 donnée dans [WBSS04]. Sur cette figure, la luminance croît avec la SSIM sur la carte correspondante, alors qu'elle décroît avec les différences en valeurs absolues, afin de pouvoir comparer les méthodes. On s'aperçoit que la SSIM réagit à la perte des arbres au premier plan avec de faibles notes (zones sombres).

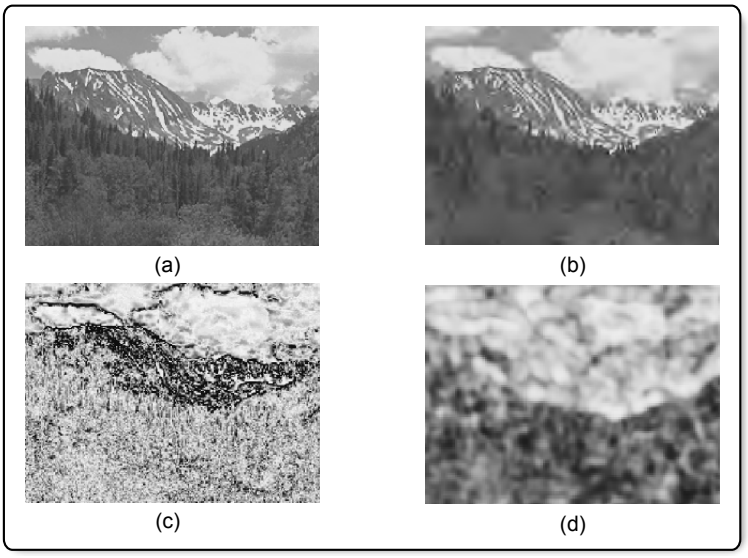


Figure 2.15 – *Cartes de distorsions. a) image source, b) image compressée, c) différences absolues, pixel à pixel, d) carte SSIM*

Des optimisations ont été apportées par la suite à cette métrique. Les mêmes auteurs ont proposé une version multi-échelle (MS-SSIM) dans [WSB03]. Comme pour la synthèse multi-résolution, les niveaux supérieurs de la pyramide sont des versions sous-échantillonnées par un filtrage passe-bas. Après avoir calculé indépendamment les termes l, c et s pour chaque étage, la SSIM finale est donnée par :

$$MS-SSIM(x,y) = [l(x,y)]^{\alpha_M} \cdot \prod_{j=1}^{M} [c(x,y)]^{\beta_j} \cdot [s(x,y)]^{\gamma_j} \tag{2.15}$$

où M est le nombre d'étages de la pyramide, le paramètre α_M dépend du nombre d'étages et les paramètres β_j et γ_j dépendent du niveau de résolution courant. l, c et s correspondent ici au cumul sur un niveau de résolution des mesures sur chaque fenêtre glissante f précédemment introduite.

Une autre extension est décrite dans [WLB04] pour prendre en compte l'aspect temporel de la perception des séquences vidéo. La principale contribution consiste à augmenter l'importance des zones claires de l'image, qui, pour les auteurs, sont censées attirer le regard. Pour cela, une pondération est appliquée par fenêtre suivant la luminance moyenne μ_f suivant :

$$w_f = \begin{cases} 0 & \mu_{f_x} \leq 40 \\ \frac{\mu_{f_x}-40}{10} & 40 < \mu_{f_x} \leq 50 \\ 1 & \mu_{f_x} > 50 \end{cases} \tag{2.16}$$

Une pondération suivant le mouvement global est aussi appliquée pour prendre en compte le fait que lors d'un mouvement rapide, l'œil perçoit moins bien les distorsions de structure. Ainsi, si le module moyen M des vecteurs mouvements de la scène est élevé, une pondération faible sera appliquée afin de valoriser les moments de la séquence où le mouvement est faible ou nul, moments ou l'œil peut fixer et détecter le maximum de distorsions.

Cette pondération suit pour chaque image i :

$$W_i = \begin{cases} \sum_{j=1}^{R} w_{f_j} & M_i \leq 0.8 \\ \frac{1.2-M_i}{0.4} \sum_{j=1}^{R} w_{f_j} & 0.8 < M_i \leq 1.2 \\ 0 & M_i > 1.2 \end{cases} \tag{2.17}$$

avec R le nombre de fenêtres f_j sur l'image et M_i le module moyen du mouvement défini par

$$M_i = \frac{\sum_{j=1}^{R} m_{i,f_j}}{RK_M}, \tag{2.18}$$

où m_{i,f_j} correspond à chaque vecteur de mouvement pour les fenêtres considérées sur l'image i et K_M une constante de normalisation fixée à $K_M = 16$ par les auteurs. Ces adaptations au mouvement sont cependant approximatives par rapport aux propriétés du SVH énoncées en sections 2.1.2, 2.1.3 et 2.1.4. Premièrement, ce sont les contrastes et non les luminances moyennes qui influent le plus sur la perception. D'autre part, si l'idée de la prise en compte des mouvements rapides appliquée sur l'image entière s'avère intéressante, une approche locale pour chaque objet serait peut-être plus pertinente. Quoi qu'il en soit, cette métrique prend en compte l'aspect temporel, alors que le PSNR omniprésent dans le domaine de la compression de vidéo n'en tient nullement compte.

Méthodes orientées SVH.

Les métriques orientées SVH prennent en compte, à leur manière, les notions de masquage et de facilitation évoquées en début de chapitre. Le critère VDP (Visible Difference Predictor), présenté dans [Dal93], propose par exemple d'évaluer la probabilité que le SVH détecte une différence entre une image dégradée et celle d'origine. Cette méthode utilise une décomposition en ondelettes. Dans chaque sous-bande considérée, des cartes de seuils de détection sont calculées, puis cumulées sur l'image pour obtenir une note finale de qualité.

Une autre métrique, décrite dans [OLL+03], propose d'abord de détecter les artefacts sur la luminance. L'image originale et l'image dégradée sont d'abord filtrées par un filtre passe-haut. Pour chaque position (x, y), entre un signal $s(x, y)$ et sa version dégradée $s'(x, y)$, la probabilité que la différence soit détectée est donnée par

$$p(x,y) = \begin{cases} 1 & si\ |s(x,y) - s'(x,y)| > e_1 \\ 0 & si\ |s(x,y) - s'(x,y)| < e_0 \\ \frac{|s(x,y)-s'(x,y)|-e_0}{e_1-e_0} & \text{sinon} \end{cases}, \tag{2.19}$$

où e_0 et e_1 sont des seuils dont e_0 le seuil en dessous duquel la différence est considérée comme invisible. Un gain est ensuite appliqué en prenant en compte les aspects temporels, de fréquences spatiales et de contraste. De plus, une évolution permettant la prise en compte des chrominances est présentée dans [OLLY05].

2.5.4 Conclusion sur les métriques.

D'autres métriques ont été développées, tentant de modéliser au mieux l'appréciation des différences par le SVH. Elles permettent cependant d'évaluer une différence entre les pixels de la séquence dégradée par rapport à la source. Dans le cadre d'un schéma qui utilise la synthèse de texture, ces métriques ne permettent pas d'évaluer la qualité de la reconstruction puisque les différences entre l'image synthétisée et sa source peuvent

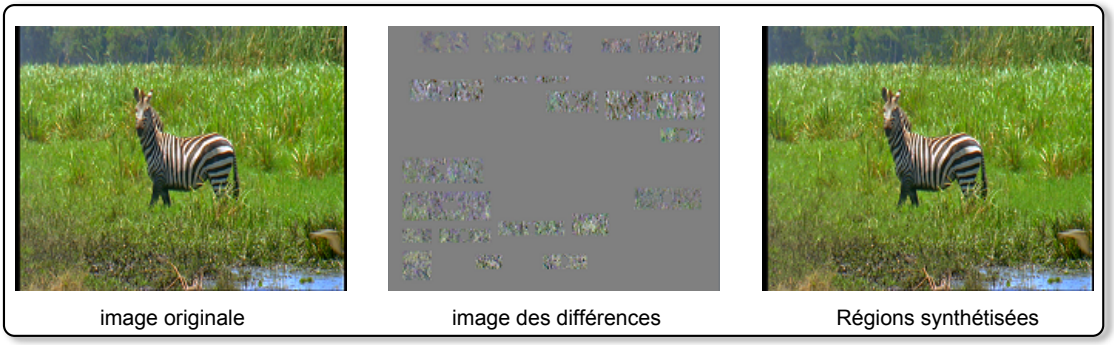


Figure 2.16 – *Régions reconstruites par synthèse, et donc à fortes différences.*

être importantes, alors que la région synthétisée parait similaire à l'œil. Il n'existe, à ce jour, pas de métrique qui permette de juger réellement des défauts que l'œil humain pourrait percevoir, sur une région reconstruite de cette manière. La figure 2.16 présente le cas classique de reconstruction visé par ces travaux de thèse. L'image des différences montre bien qu'il est difficile d'évaluer si l'image est de bonne qualité. En effet, vis-à-vis de l'image source, de grandes différences apparaissent, même en prenant en compte les aspects du SVH cités précédemment. Des tentatives d'évaluation de qualité sans référence à la source ont été tentées. Elles restent cependant liées à un contexte particulier avec une connaissance *a priori* sur les artefacts à chercher.

2.6 De nouveaux outils.

2.6.1 Choix des modes de prédiction et distorsions associées

Les mesures de distorsion dans le standard H.264, telles que le PSNR ou les calculs d'erreur pour les choix de mode, sont fondés sur le calcul moyenné de distances pixel à pixel. Les propriétés du SVH ne sont donc nullement prises en compte, les choix de modes de compression sont donc perfectibles.

Les principales contraintes dans la réalisation des normes de codages de vidéo résident dans la complexité du décodeur. En effet, une intelligence minimale a toujours été requise afin d'être capables de décoder la vidéo en temps réel, quels que soient l'application et le matériel : décodeurs pour télévision, smartphones, tablettes... Avec la progression de la rapidité de traitement des systèmes embarqués, il est cependant maintenant possible de transférer de l'intelligence au décodeur afin d'accroître considérablement les performances de codage, chose extrêmement difficile en bloquant ce degré de liberté, les codeurs H.264, étant très aboutis en termes de performances de codage.

2.6.2 Le *Template Matching* pour la prédiction intra

Une nouvelle recherche de prédicteurs en mode intra appelé *template matching* est proposée dans [TBS06]. Illustré par la figure 2.17, ce mode de prédiction cherche à faire correspondre une zone *template* constituée des pixels causaux du voisinage direct du bloc à prédire avec les pixels contenus dans une zone de même forme dans la partie causale de l'image. Le bloc candidat associé à la zone template la plus corrélée à celle du bloc courant sert alors de prédicteur au même titre qu'une prédiction intra classique.

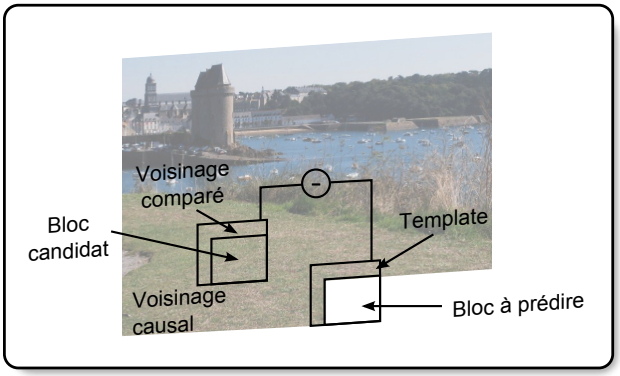


Figure 2.17 – Template matching *: Prédiction d'un bloc en fonction de son voisinage* template.

Cette approche est intéressante par le fait qu'elle établit les liens étroits tissés entre la prédiction dans le domaine de la compression et le domaine de la synthèse de texture. En effet, ce travail, qui n'est pas sans rappeler la comparaison de *voisinages* décrite dans le premier chapitre, est en partie fondé sur les approche de L.Y. Wei [WL00] et M. Ashikhmin [Ash01].

Vers le transfert de l'intelligence au décodeur

La section suivante présente les approches voisines de celle visée dans ce manuscrit. Fondées sur l'adaptation au contenu de la scène à encoder, ces méthodes passent d'abord par le transfert de l'intelligence côté décodeur. Contrainte majeure des normes de compression actuelles, les décodeurs capables de reconnaître et de synthétiser des régions de texture seront vraisemblablement beaucoup plus complexes que leurs parents utilisés actuellement.

Les codeurs orientés contenu, visent à traiter séparément des régions construites suivant diverses cohérences structurelles.

- Le mouvement : le regroupement des vecteurs de mouvement pour chaque pixel en classes permet de segmenter des régions spatio-temporellement cohérentes.
- Les textures : les outils de segmentation et de description de texture sont variés et feront l'objet d'une étude approfondie dans le chapitre 3.

2.7 Les schémas adaptés au contenu.

2.7.1 Motivations.

Le standard H.264 [AVC03] est la référence de compression dans l'état de l'art actuel. Le processus de prédiction opéré dans ce standard est fondé sur un compromis entre débit et distorsion présenté dans [WSJ+03]. D'un point de vu compression objective des données, cette méthode produit des calculs de prédiction de bonne qualité. Cependant, la distorsion utilisée, fondée sur le critère de la MSE, ne permet pas toujours une corrélation adéquate entre les modes choisis et les caractéristiques du système visuel humain.

Deuxièmement, les textures sont souvent des surfaces très détaillées et bruitées pour lesquelles les opérations de quantification dans un domaine transformé en fréquences spatiales

ne sont pas adaptées. La quantification dans le domaine fréquentiel sacrifie typiquement les coefficients de hautes fréquences, qui contiennent les informations de détails.

Une autre limitation, décrite dans [NNBW09a], provient de considérations temporelles puisque seule la compensation de mouvement, avec plusieurs images de référence, permet de tenir compte des statistiques à long terme dans les séquences vidéo. Il est alors probable qu'une meilleure adaptation temporelle, à l'évolution du contenu permettrait d'améliorer directement les performances. Les approches décrites ci-après tentent, par exemple, de reproduire des régions texturées, temporellement cohérentes sur un ou plusieurs GOP.

Les schémas présentés dans cette section partent donc du postulat que certaines textures n'ont pas besoin d'être analysées par le SVH. Qualifiées de textures *non pertinentes*, elles peuvent être reconstruites d'une autre manière que par des techniques fondées sur la MSE : les outils de synthèse de texture.

Des outils existent pour les étapes de synthèse et d'analyse. Les algorithmes de synthèse présentés dans le chapitre 1 sont ainsi largement utilisés. D'autre part, des outils d'analyse existent déjà. Le standard MPEG-7, présenté dans [MSS02], contient de nombreux descripteurs permettant de caractériser des informations visuelles. Ces informations sont cependant dédiées à la description de contenus multimédia à des fins de stockage, de référencement, et de facilitation de communication. A l'inverse, les schémas de codage présentés dans cette partie visent à optimiser l'utilisation de la description du contenu pour réduire le débit de données à transmettre en préservant la qualité visuelle. L'approche présentée dans [BZD07, BZD08] est par exemple focalisée sur la qualité de l'analyse spatio-temporelle des textures côté codeur.

2.7.2 Approche multi-couches.

Un des premiers travaux de compression adaptée au contenu a été présenté dans [Ade91]. Cette approche traite la séquence vidéo comme des couches qui se chevauchent ou se recouvrent, définissant des régions ayant un mouvement cohérent. Ainsi, cette analyse se focalise ici sur la cohérence en mouvement et non sur la texture. Une étape de segmentation permet d'extraire ces régions et fournit un panel de cartes permettant de les détourer. Les travaux présentés ensuite dans [WA94] proposent une approche avec 4 cartes de segmentation suivant :

- *l'intensité lumineuse*,
- le *canal alpha* qui correspond au degré d'opacité d'une couche (de complètement transparent à opaque),
- la *vélocité* du mouvement pour chaque pixel,
- et la *carte delta* qui correspond aux variations temporelles de luminance sur chaque position.

Ce schéma est par ailleurs qualifié d'*open loop* ou en boucle ouverte du fait qu'il n'existe pas de mécanisme de retour capable d'identifier les artefacts sur les régions où la segmentation est défaillante. Il n'y a donc pas de moyen de compenser une segmentation défaillante a posteriori. Il en résulte que la qualité des vidéos décodées n'est pas contrôlée et qu'il est difficile de déterminer les performances d'un tel schéma.

2.7.3 Approches paramétriques.

Première approche

Ces travaux, présentés d'abord dans [DHIC03], puis de manière plus exhaustive dans [DH04], décrivent un système dans lequel les textures remplaçables sont détectées puis retirées du côté du codeur et synthétisées au décodeur. Une texture potentiellement synthétisée est définie au codeur comme une région texturée qui apparait dans au moins 40% d'un GOP de 50 images. Le codeur contient donc une phase d'analyse comprenant une première étape de segmentation et une analyse spectrale des textures segmentées. Ensuite le décodeur synthétise les régions manquantes grâce à l'approche proposée dans [PS00a]. Les auteurs prétendent que ce schéma peut préserver 55% de débit à qualité visuelle semblable, comparé à H.264, sur certaines séquences dont toutes les images considérées sont codées en mode intra.

Cette approche pragmatique contraste avec l'approche présentée précédemment. En effet, seules les régions considérées comme synthétisables sont traitées, et l'analyse laisse ainsi les autres régions à un processus d'encodage classique, *i.e.* utilisant un codeur standard de l'état de l'art.

Schéma paramétrique utilisant un modèle autorégressif pour la synthèse de texture.

Ce schéma, présenté dans [KGL+08], diffère des autres approches orientées synthèse de l'état de l'art par le fait qu'une approche paramétrique est utilisée. Ce modèle autorégressif sert non seulement à la synthèse mais aussi à la caractérisation des textures. Les auteurs assurent que cette technique permet de retirer au codeur des régions plus larges par rapport à l'utilisation de méthodes d'inpainting, tout en gardant une qualité visuelle équivalente. Cette propriété permet ainsi de réduire d'autant plus le débit nécessaire à la transmission. La détection des blocs potentiellement supprimés, puisque synthétisables, repose aussi sur un seuillage des gradients calculés pour un bloc par :

$$\sum_{bloc} \sqrt{G_x^2 + G_y^2} < Th \tag{2.20}$$

avec G_x et G_y représentant l'énergie des gradients horizontaux et verticaux en chaque pixel du bloc, calculés via l'opérateur de Sobel. Th est un seuil déterminé par l'utilisateur : si l'énergie sur le bloc le dépasse, le bloc est considéré comme structurel et nécessite d'être classiquement encodé et transmis par le codec H.264 dans les expérimentations proposées. Le schéma est testé sur des GOP contenant seulement des images I et P. Alors que les images I sont encodées en intra par H.264, les images P sont modélisées par le modèle autorégressive spatio-temporel.

Le modèle paramétrique permet de meilleures performances que l'algorithme présenté dans [ZSWL07] pour les hauts débits selon les auteurs. Cependant, les images présentées en résultats visuels, sur la figure 2.18, ne permettent pas d'évaluer clairement la qualité des zones synthétisées.

Les sections suivantes présentent des schémas de compression utilisant les standards actuels ou futurs, en y ajoutant une étape au codeur permettant de retirer les textures afin de diminuer le débit.

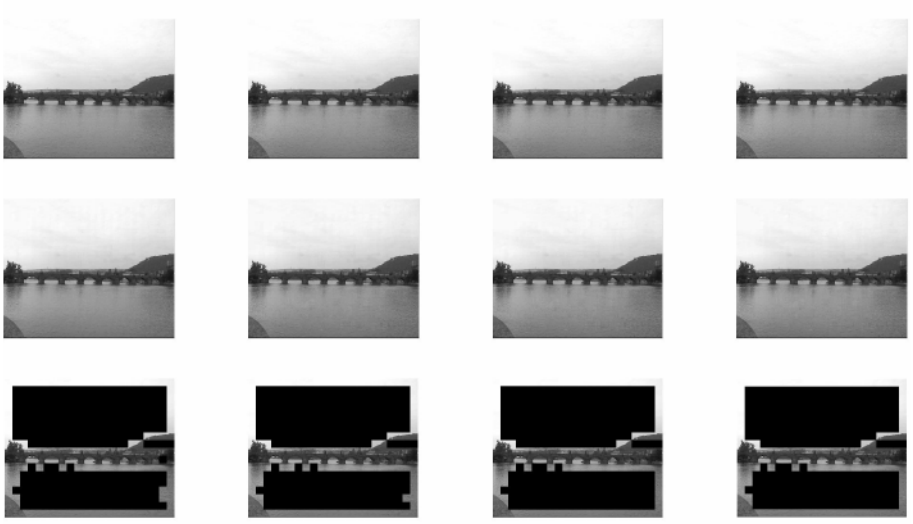


Figure 2.18 – *Résultats fournis dans [KGL+08] pour la séquence* Bridge far. *La ligne du haut montre les images décodées par le standard H.264, la ligne du milieu présente la reconstruction avec l'approche proposée et la dernière ligne montre les images originales avec les blocs retirés. Les résultats sont présentés avec QP=12*

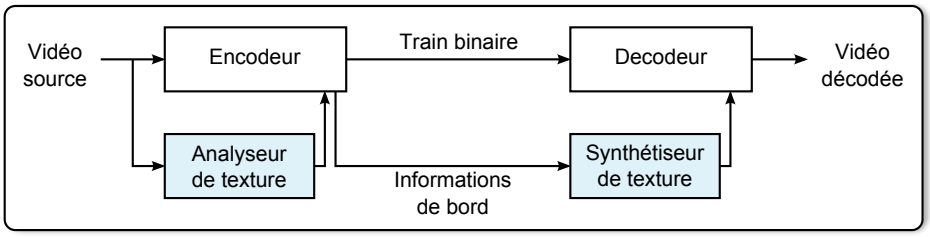


Figure 2.19 – *Schéma de codage orienté synthèse de texture proposé par P. Ndjiki Nya.*

2.7.4 Schéma générique de compression orienté synthèse.

Les premiers travaux de codeurs orientés perception de la texture développés par P. Ndjiki-Nya et présentés dans [NNMS+03, NNMB+03] datent de 2003. Ils définissent un schéma global de système de codage, illustré en figure 2.19, comportant des outils d'analyse de texture côté codeur, et de synthèse côté décodeur. Cette structure a été largement reprise dans la littérature. L'analyseur de texture présenté en figure 2.19 contient une étape de segmentation fondée sur la fusion de blocs considérés comme homogènes. Un *quadtree* permet d'abord de diviser l'image suivant un critère de similarité décrit plus loin. L'image est alors séparée en 4 grands blocs distincts. Pour chacun de ces blocs, si les 4 blocs fils à l'étage suivant ont des propriétés statistiques équivalentes, alors le bloc est considéré comme homogène. Dans le cas inverse, l'opération continue sur les 4 blocs fils, et ainsi de suite. Une fois ces opérations terminées, les blocs voisins sont comparés avec le même critère, et sont fusionnés si considérés comme similaires. Ces opérations se terminant à la stabilité du système. La mesure de similarité repose sur l'utilisation de descripteurs fournis par la norme MPEG-7 [MSS02] : le descripteur EH pour *Edge Histogram* et SCC pour *SCalable Color*. Le premier relève les contours suivant les directions horizontale, verticale et diagonale alors que le second correspond à un histogramme de couleurs.

Contrairement aux travaux de C. Zhu [ZSWL07] présentés dans la section 2.7.7 , l'approche est testée en complément du codeur H.264. Cependant les contributions se situant en dehors du codec, elles sont génériques et peuvent être compatibles avec d'autres systèmes permettant d'encoder et décoder les blocs transmis.

Les besoins d'évaluation de la méthode, utilisée pour reconstruire les régions, ont conduit l'auteur à créer une métrique adaptée, présentée dans la section suivante.

2.7.5 Une métrique pour attester de la qualité de la synthèse.

Construction de la métrique.

Après avoir construit un premier schéma incluant analyse/synthèse de texture dans [NNMS+03], P. Ndjiki-Nya a proposé dans [NNKW04] de définir une métrique appelée VQA (Video Quality Analysis) qui permet de détecter les artefacts de leur synthèse. Cette métrique est fondée sur un outil de détection de contours. Les filtres horizontaux et verticaux des gradients de Kirsh sont utilisés afin de détecter des frontières verticales ou horizontales, créées par l'algorithme permettant d'envelopper la texture choisie sur la région courante. Il permet ainsi de détecter les frontières rectilignes, potentiellement crées par l'ajout de blocs de texture, dont le « collage » avec le contour n'a pas fonctionné.

Évolution du schéma par l'utilisation de la VQA.

Pour le premier schéma présenté dans [NNKW04], seules les textures rigides sont considérées. Le meilleur candidat pour la texture manquante est détecté dans l'image de référence au codeur par la mise en correspondance avec un critère MSE. Les informations de position et de mouvement minimisant ce critère sont transmises par le codeur afin que le décodeur puisse retrouver, appliquer et envelopper la région candidate à l'endroit de la texture manquante. Comme dans les travaux de C. Zhu, seules les images B des GOP utilisés par H.264 sont potentiellement traitées par les contributions apportées. Les auteurs prétendent préserver jusqu'à 19.4% de débit pour un niveau de qualité subjectif similaire, ce gain diminuant aussi à mesure que QP augmente.

Les travaux présenté dans [NNHSW05] font état de l'ajout d'une analyse au codeur, qui permet de séparer les textures *pertinentes* pour le SVH des *non-pertinentes*. Cette propriété est définie par l'étude des mouvements de la région considérée. Un algorithme utilisant une M-estimation permet de classifier les régions animées d'un mouvement homogènes, qui pourront être décrites avec des paramètres de translation, rotation et facteur d'échelle en vue de leur synthèse au décodeur. Ainsi, seules les textures dites *rigides* sont ensuite traitée, *i.e.* retirée par le codeur et synthétisées au décodeur. Une métrique VQA temporelle permet d'attester de la bonne cohérence temporelle des textures synthétisées. Une machine d'état sert ensuite à tester plusieurs paramètres de synthèse, en utilisant la métrique VQA, afin de d'identifier ceux qui permettent d'aboutir à la meilleure synthèse.

2.7.6 Compression d'images fixes utilisant des techniques d'inpainting.

Un schéma de compression alliant analyse de texture et méthodes d'inpainting a été présenté dans [WSWX06]. Ces travaux sont fondés sur des approches précédentes dédiées à reconstruire des images détériorées lors de la transmission. Le but était alors de *boucher* des blocs perdus à partir de leur voisinage, ces méthodes étant connues sous le nom d'*error concealment*. Le schéma présenté dans [RSB02] par exemple, préfigurait des futurs travaux orientés compression. Il prévoyait déjà de dissocier structure et texture à partir des blocs

voisins de la région manquante. Une fois cette distinction faite, un algorithme d'inpainting inspiré de [BSCB00] permettait de propager les structures, alors que l'algorithme de synthèse orientée pixel remplissait les blocs considérés comme texturés.

Lorsque des structures sont présentes dans les régions enlevées, le codeur envoie une information supplémentaire sous forme de masque binaire. Ce dernier marque les contours et permet, côté décodeur, de reconstruire les formes grâce à un algorithme d'inpainting dédié. L'outil utilisé pour cette détection de contour s'appelle EDISON pour Edge Detection and Image SegmentatiON [GC03]. Ensuite, après la propagation du signal le long des contours définis par le masque binaire, les auteurs ont choisi une synthèse orientée patch afin de synthétiser les pixels inconnus, alors considérés comme texture. Pour ce faire, les algorithmes issus de [EF01] et son optimisation dans [KSE+03] ont été choisis afin d'éviter d'introduire des zones de flou, principal reproche fait aux algorithmes orientés pixel. L'outil général de reconstruction de région parait complet pour la compression de certaines régions et donne des résultats visuellement prometteurs. Il reste néanmoins une étape cruciale qui n'est pas détaillée côté codeur : la sélection des régions encodées et celles qui seront reconstruites au décodeur. En effet, une segmentation prenant en compte la couleur, des statistiques de premier ordre et une intervention de l'utilisateur est annoncée avec les résultats. Ainsi, la décision d'encodage ou non des blocs n'est aucunement liée avec la méthode de reconstruction.

Cette faille est abordée dans les travaux présentés dans [LSW+07]. En effet, l'étape d'extraction des contours sert maintenant aussi à l'étape d'analyse d'image côté codeur pour la décision à faire à propos des régions qui seront encodées classiquement ou non. Le détecteur de contour présenté dans [RMHN95] sert ainsi à détailler les régions potentiellement retirables, puisque reconstructibles par les outils d'inpainting et de synthèse. Il sert aussi à définir des blocs dans ces régions qui ne seront classiquement encodées puis décodées quand les zones manquantes seront trop larges, ces zones seront alors reconstruites par les méthodes d'inpainting usuelles. Ce schéma permet d'après les auteurs de préserver 44% de débit par rapport à la compression opérée par JPEG avec une qualité visuelle similaire, et 33% par rapport à H.264 en mode intra.

Malgré les résultats chiffrés prometteurs pour ce type d'approche, les résultats en images proposés montrent qu'une quantité importante de blocs à l'intérieur des régions retirées sont classiquement encodés, afin de garder la cohérence spatiale des couleurs. Une optimisation en ce sens est proposée dans [XSW10] : elle introduit des paramètres vectoriels permettant de définir les régions dont la couleur ou l'intensité change de manière graduée, afin de guider les algorithmes classiques d'inpainting utilisés. La figure 2.20 illustre les régions enlevées ainsi que les contours en bleu qui permettent de guider l'inpainting.

2.7.7 Vers une approche utilisant de la synthèse spatio-temporelle.

Approche de C. Zhu

Ces travaux présentés dans [ZSWL07], procèdent aussi en enlevant des régions côté codeur pour les synthétiser côté décodeur, et ainsi préserver le débit qui aurait été nécessaire à la transmission de l'intégralité des régions. Cette méthode est intégrée au cœur d'un schéma compatible avec le standard H.264. Ainsi, dans les GOP prévus dans H.264 qui sont considérés, seules les images B sont potentiellement traitées par la synthèse de texture. Aussi, cette approche considère des blocs de taille 8×8. Premièrement une étape de segmentation permet de classifier ces blocs en deux catégories :

- les *blocs de structure* qui contiennent les contours et les objets de l'image,

Figure 2.20 – *Résultats fournis dans [LSW+07] Les images de gauche présentent les blocs enlevés ainsi que l'information de contour envoyée comme information supplémentaire pour l'inpating (en bleu) ; au centre les images reconstruites par les outils présentés ; à droite les images reconstruites par JPEG avec le profil de base.*

- les *blocs de texture* : le reste des blocs qui contiennent de la texture. Ils sont potentiellement enlevés au codeur pour être synthétisés au décodeur.

Cette segmentation est fondée sur un simple détecteur de discontinuités par seuillage. Un algorithme de synthèse de texture orienté patch est utilisé pour la synthèse de texture. Inspiré des travaux présentés de V. Kwatra dans [KSE+03], ce schéma ne contient cependant pas toutes les optimisations de l'outil *Graphcut*, et se trouve aussi limité dans la recherche spatiale des patchs utilisés. En effet, seuls des blocs 8×8 peuvent être considérés. Aussi, pour éviter les potentielles inconsistances temporelles au niveau des régions synthétisées, une étape d'estimation de mouvement permet une meilleure sélection des patchs utilisés pour la synthèse dans les images de référence.

Afin d'obtenir une forte cohérence temporelle, les auteurs exploitent la technique des motion threads, définis ci-dessous, et illustrés sur la figure 2.21. Du point de vue du *block matching*, défini en section 2.4.2, un macrobloc d'une image B_i (par exemple B_1 sur la figure 2.21), possède toujours un prédicteur temporel dans une image de référence donnée.

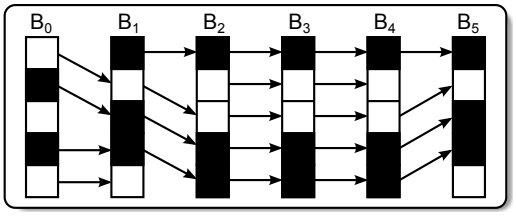


Figure 2.21 – *Motion threading*

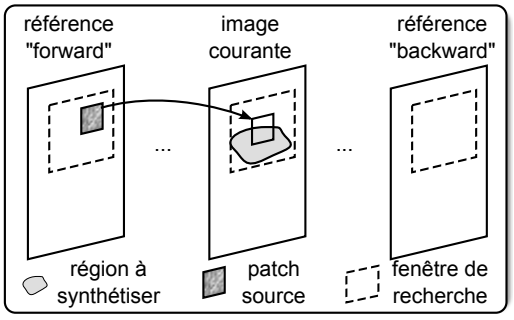


Figure 2.22 – *Schéma spatio-temporel de synthèse de texture*

Ce dernier correspond au macrobloc qui lui ressemble le plus parmi tous ceux de l'image de référence : l'image B_0. Ce prédicteur possède lui-même un prédicteur dans une autre image de référence (l'image B_2). De proche en proche, on définit ainsi une série de prédicteurs, cette série définissant un *Motion Thread* (MT).

Il est décidé de coder non pas un bloc texturé seul, mais tout le MT auquel il appartient. La solution de [ZSWL07] propose de ne préserver que ceux qui ont une variation spatio-temporelle importante, *i.e.* ceux qui sont composés de blocs très différents de leurs voisins spatiaux et temporels (au dessus, en dessous, à droite, à gauche, colocalisés avant et après) ou possédant une variance élevée. La variation spatio-temporelle (VST) d'un MT est calculée par

$$VST = \frac{1}{N}\sum_{i=0}^{N-1}\left(w_1\delta(B_i) + w_2 \sum_{B_j \in \mu_6(B_i)} |E(B_j - E(B_i)|\right) \tag{2.21}$$

où N est la longueur du MT, les B_i sont les blocs qui le composent et les B_j sont les blocs contenus dans le voisinage spatio-temporels μ_6 de B_i. $\delta()$ désigne la fonction de variance et $E()$ la moyenne d'un bloc, w_1 et w_2 sont des facteurs de pondération. La figure 2.22 présente les références possibles pour un région à synthétiser.

Enfin, parmi les blocs texturés restants, afin d'éviter que de trop grandes zones connexes ne soient supprimées, il est nécessaire de préserver certains blocs texturés régulièrement espacés à l'intérieur des zones à supprimer. Ces blocs sont illustrés sur la ligne du haut de la figure 2.24.

Le patch candidat le plus similaire est celui qui possède la valeur de similarité S la plus faible, donnée par

$$S = SSD(W_t, W_c) + \alpha \cdot SSD(W_{t'}, W_c). \tag{2.22}$$

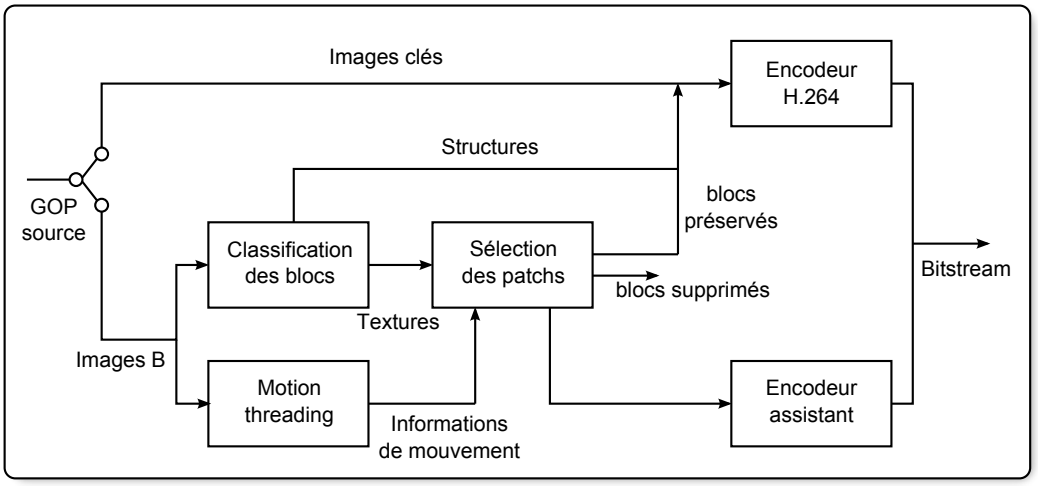


Figure 2.23 – *Schéma du codeur présenté dans [ZSWL07]*

S est ainsi définie comme la somme pondérée de deux sommes des différences des pixels connus élevées au carré entre deux patchs (SSD pour Sum of Squared Differences). W_t correspond au patch cible, W_c au patch candidat, $W_{t'}$ représente le patch colocalisé dans l'image de référence et α le facteur de pondération. L'article reste flou dans la technique exacte permettant la suite de la synthèse, c'est à dire la méthode d'ajout du patch candidat choisi, en citant uniquement l'algorithme présenté dans [KSE+03] ainsi que l'outil développé dans [PGB03]

Approche de P. Ndjiki-Nya.

Les travaux présentés dans [NNSW05] prennent aussi le parti d'exploiter la méthode des graphcuts, introduite dans [KSE+03], pour la synthèse de textures. Le schéma se limite ici à la reconstruction des zones temporellement homogènes. La technique des graphcuts permet de réduire la visibilité des frontières avec les blocs avoisinant précédemment décodés. Pour parfaire la transition, P. Ndjiki-Nya utilise une synthèse multirésolution, utilisant une pyramide Laplacienne. Cette dernière est appliquée à la zone de chevauchement entre zone ajoutée et zone décodée, puis synthétisée de la version la plus grossière à la plus fine, afin de filtrer les hautes fréquences potentiellement introduites lors d'une synthèse monorésolution.

Dans ce schéma, les régions de plusieurs images B successives sont potentiellement retirées au codeur, du moment que plusieurs GOP, avant et après, sont classiquement encodés par H.264. Pour gagner un maximum de débit sur ces blocs, aucune information ne doit être transmise. Dans le standard H.264, lorsqu'un MB possède un mouvement qui peut être efficacement prédit par ceux des MB voisin, et qu'il ne contient aucun coefficient résiduel transformé et quantifié non nul, alors il lui est appliqué le mode *skip*. Aucune information n'est transmise, si ce n'est l'information du mode : le décodeur reconstruit le MB à partir de l'image de référence et des vecteurs de mouvement de ses voisins. Les auteurs proposent d'étiqueter les blocs à synthétiser comme *skippés* afin de ne transmettre ni résidu ni informations de mouvement. Seul un bit est nécessaire afin de distinguer ce mode dédié aux MB à synthétiser du mode naturel d'H.264.

Un alignement temporel peut aussi être effectué, dans le cas d'un déplacement de

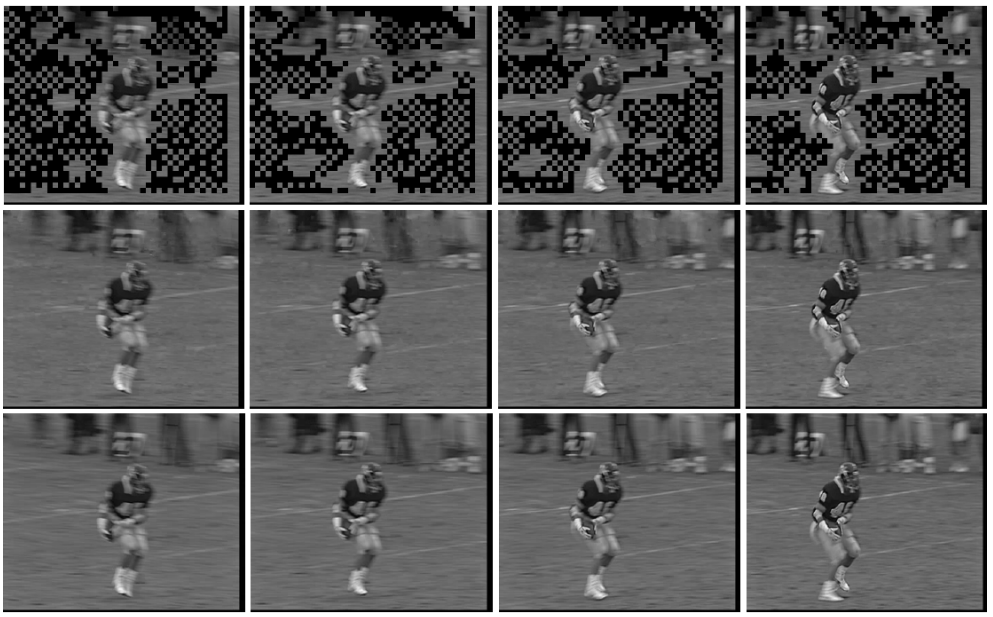


Figure 2.24 – *Résultats fournis dans [ZSWL07] pour la séquence* Football. *4 images B sont présentées : la ligne du haut montre les images originales avec les blocs retirés, la ligne du milieu présente la reconstruction avec l'approche proposée et la dernière ligne montre les images décodées par le standard H.264. Les résultats sont présentés avec QP=18*

caméra. Dans ce cas, les paramètres de déplacement sont estimés au codeur et envoyés comme information supplémentaire pour une synthèse temporellement cohérente au décodeur. Ce schéma, malgré le fait qu'il se limite aux textures rigide, apporte les bases des solutions performantes, décrites dans la section 2.7.9

2.7.8 Utilisation de la synthèse EM.

Contrairement aux autres approches, le schéma présenté dans [OSS+08] utilise un synthétiseur orienté pixels et non patchs. En effet, le processus de décodage intègre l'algorithme proposé dans [KAK05] (section 1.7.2), suivant une approche EM. Une autre différence majeure se situe dans le traitement des régions de texture, qui ne sont plus intégralement supprimées. Une version sous-résolue est encodée et transmise au décodeur, l'information envoyée pour ces régions est ainsi constituée d'une version fortement quantifiée par le codeur H.264. Un budget de débit est alloué à une région de texture afin de transmettre cette version sous-résolue comme information supplémentaire au décodeur. Dans les faits, la qualité de ce signal est, dans les expérimentations, directement pilotée par l'ajustement du QP du codeur type H.264. Cette information sert ensuite, côté codeur, à initialiser la surface à synthétiser par l'algorithme EM multi-résolution. Malheureusement, cette idée prometteuse n'est pas suivie par une partie résultat permettant d'en apprécier l'efficacité. Les résultats sont présentés sur de petits patchs, illustrés en figure 2.25, du fait qu'aucune segmentation n'est proposée pour le processus d'encodage.

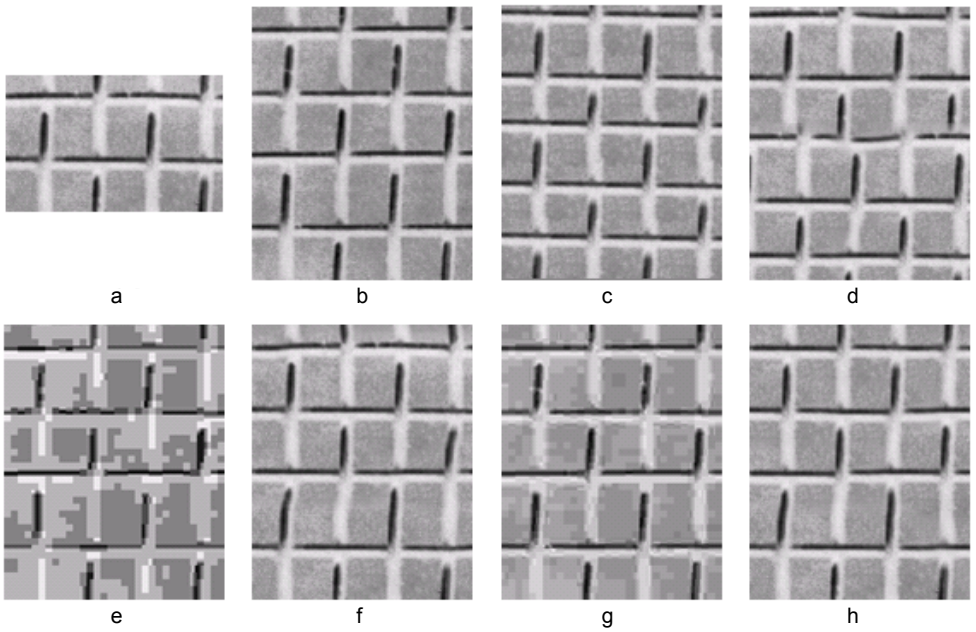


Figure 2.25 – *Résultats obtenus par la synthèse de Byung Tae Oh. l'image (a) montre le patch source retenu pour la synthèse, décodé avec $QP = 20$, (b) l'image source à reproduire par la synthèse, (c) le résultat de l'approche de P. Ndjiki-Nya dans [NNHW07], (d) texture synthétisée sans information supplémentaire, version quantifiée à $QP = 50$, (f) synthèse en utilisant (e) comme initialisation de la synthèse EM, (g) $QP = 40$, (h) synthétisé en utilisant (g)*

2.7.9 Approches séparant textures rigides et déformables.

Approche de C. Zhu

Les auteurs de [ZSWL07] proposent une approche plus robuste séparant le traitement des régions animées de mouvements globaux et locaux. Comme dans la première approche, dans les GOP considérés, les images I et P servent d'images clés alors que les images B sont divisées en blocs 8×8 de structures qui pourront être classiquement encodés, et les autres qui seront synthétisés au décodeur.

- Dans le cas des régions animées de mouvements locaux, la sélection des blocs supprimés et de ceux utilisés comme patch source pour la synthèse au décodeur suit la méthode de leurs précédents travaux détaillés dans la section 2.7.7. De même, côté décodeur, la synthèse ne diffère pas de l'approche précédente.
- Pour les régions suivant le mouvement global sur le GOP, un *sprite* est généré à la manière de [LGW03]. Un sprite correspond à la mise à plat, bout à bout, des images d'un GOP. De la même manière que pour la sélection des blocs de structure dans [ZSWL07], les blocs du sprite sont ensuite divisés en blocs de structures et blocs de structures. Cependant, les blocs de structure et les blocs de texture peuvent être considérés comme patchs sources pour la synthèse dans ce cas. En effet, comme une estimation globale de mouvement est appliquée, le même modèle de mouvement est utilisé pour les structures et les textures. La sélection des patchs sources suit alors celle présentée dans [LSW+07]. Du côté décodeur, la synthèse proposée dans [LSW+07] est utilisée pour remplir les régions manquantes des sprites reconstitués par GOP.

Afin que le décodeur puisse trier les régions et obtenir les informations sur les patchs source et le mouvement global, les informations supplémentaires envoyées dans le train binaires contiennent :

- le masque des régions,
- le masque des blocs retirés au codeur,
- les paramètres du mouvement global sur l'image,
- le masque des contours sur le sprite en cours.

Les auteurs soulignent qu'avec cette approche, une réduction de plus de 35% de débit est observée par rapport au standard H.264 pour le codage de la séquence "Stephan" pour un niveau de qualité visuel similaire. Ce gain diminue au fur et à mesure que le paramètre de quantification augmente puisque l'information supplémentaire est envoyée de la même manière et sans perte.

Approche de P. Ndjiki-Nya.

Ce schéma, présenté dans [NNHW07], intègre un bloc d'analyse et synthèse de texture en boucle fermée. Comme dans [ZSWL07], des GOP entiers sont considérés pour la synthèse spatio-temporelle. Chaque région potentielle, présente dans le GOP, est analysée et synthétisée. Comme dans [ZSWL07] seules les images B sont candidates pour l'encodage orienté synthèse.

Les textures détectées sont ensuite classifiées en deux catégories distinctes, qui seront traitées par deux synthétiseurs différents, à savoir :

- Les **textures rigides**, qui sont animées d'un mouvement global. Pour ces régions, le synthétiseur est similaire à l'algorithme de compensation globale de mouvement (ou GMC pour Global Motion Compensation), à ceci près qu'il est capable de prendre en compte plusieurs régions en mouvement, alors que la GMC ne permet que de traiter le mouvement principal dans une image, *i.e.* la région la plus vaste ayant un mouvement homogène. Le synthétiseur requiert alors comme information supplémentaire le masque de segmentation de la région en mouvement ainsi que les paramètres de mouvement et un pointeur sur l'image de référence correspondante.
- Les **textures non-rigides**, qui sont localement ou globalement déformées suivant les images du GOP. Pour ces régions, le synthétiseur est inspiré de l'approche proposée dans [KSE+03]. Plusieurs GOP avant et après le GOP considéré sont nécessaires pour la collecte de patchs 2D+t. L'information supplémentaire envoyée correspond alors au masque de segmentation, les patchs sources pour la synthèse ainsi que des paramètres de mouvement pour assurer l'alignement temporel lors du décodage.

D'après Byung Tae Oh [Oh09], cette technique de synthèse est fragilisée par l'ordre raster scan de la synthèse, pouvant introduire des artefacts de propagation de motifs non désirés. De plus, le fait que la synthèse procède par « cube spatio-temporel » de taille temporellement constante peut introduire des répétitions de motifs.

Ces arguments sont justifiés mais vérifiés par l'ensemble des schémas présentés dans cette section. Aussi, une amélioration de la synthèse est présentée dans [NNBW09b]. Ces travaux proposent d'utiliser des outils permettant de lisser et cacher les artefacts potentiellement créés par la synthèse graphcut. Ces outils proviennent de l'utilisation de clonage en respectant les équations 2D+t de Poisson et des [PGB03] dérivées 2D+t covariantes de Laplace [Geo06] sur la région synthétisée. L'efficacité de l'application de ces outils après la synthèse orientée graphcuts est illustrée par la figure 2.26.

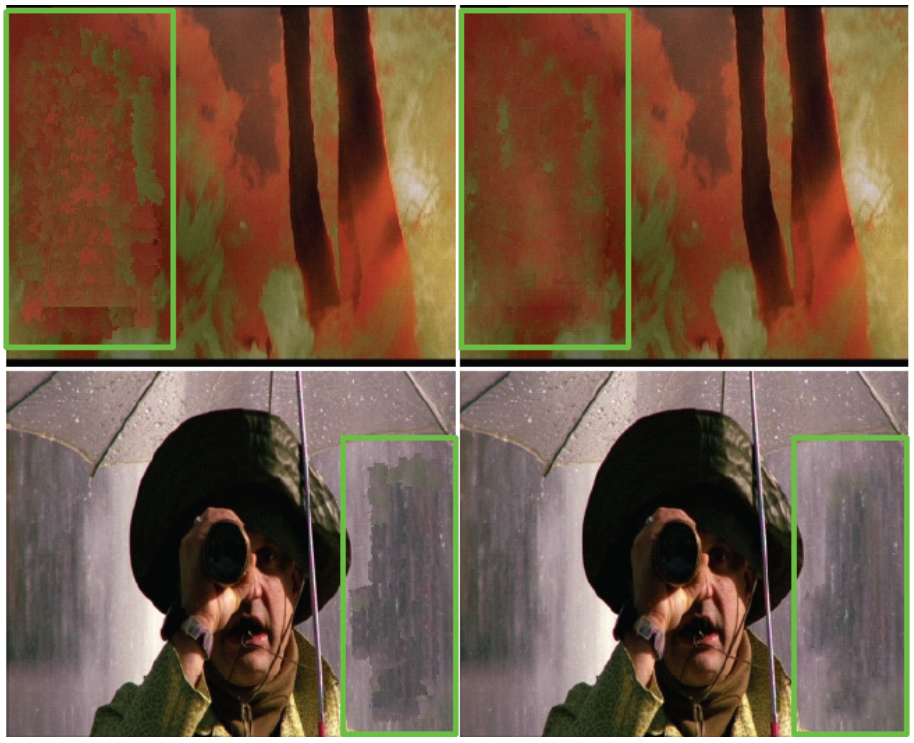


Figure 2.26 – *Lissage de la synthèse pour les séquences* Fire Flight *et* Rain. *A gauche, le graphcut sans lissage et à droite avec lissage*

Les travaux présentés dans ce papier sont focalisés sur la qualité de la texture reconstruite, les performances d'un schéma de codage ne sont donc pas spécifiquement abordées. Il est cependant acquis qu'avec un accroissement de la qualité des zones synthétisées, ces techniques peuvent permettre une nette amélioration d'encodage basé synthèse, une fois couplées avec les autres « briques » décrites dans cette section.

2.8 Conclusion sur la compression vidéo.

Les schémas hybrides, qui se sont imposés dans les standards de compression, sont toujours en phase de perfectionnement avec le développement du standard HEVC. Les contraintes de complexité et d'évaluation objective poussent les organisations de standardisation à rester sur cette ligne. Les travaux présentés à la fin de ce chapitre prouvent néanmoins que des alternatives orientées contenu sont quasi prêtes à apporter de fortes réductions de débit à qualité visuelle similaire. Fondées sur le codage séparé et adapté au type de contenu, elles permettent de traiter séparément les contours et les différents types de textures. Les textures non pertinentes pour le SVH peuvent ainsi être synthétisées au décodeur sans pour autant dégrader l'appréciation de l'œil humain. Cette dernière considération permet alors d'éviter d'encoder de larges régions détaillées, actuellement onéreuses lors d'un codage classique. Les travaux présentés dans ce manuscrit partent du même constat, dans l'optique de créer un schéma global codeur/décodeur.

Deuxième partie

Contributions

CHAPITRE 3

Segmentation et caractérisation de texture.

Les outils introduits dans la section 2.7 sont dits orientés *contenu* parce qu'ils utilisent des méthodes de codage différentes suivant les types de contenus à traiter. Les travaux présentés dans ce manuscrit nécessitent de même une analyse poussée du type de signal à encoder. Ceux-ci étant focalisés sur l'amélioration de l'efficacité de codage des parties texturées, ces dernières feront donc l'objet d'une étude approfondie. Aussi, le but de ce chapitre est de proposer une analyse cohérente des textures, contenant les deux outils fondamentaux suivant.

- Une étape de **caractérisation** : ce processus permet de déterminer certaines propriétés des textures. La taille des motifs, leur disposition, le caractère aléatoire..., sont autant de propriétés qui peuvent aider à l'algorithme de raffinement ou de synthèse de texture.
- Une étape de **segmentation** : elle vise à regrouper les pixels en régions suivant des critères de corrélation définis. L'image est alors découpée en régions cohérentes.

Ce chapitre développe les choix réalisés pour la segmentation et la caractérisation de texture, ainsi que leur adaptation aux schémas de codage développés dans ces travaux de thèse. Les deux premières sections présentent les différents types d'outils de caractérisation de texture existants, puis l'approche proposée. Les deux dernières sections sont dédiées aux travaux de segmentation.

3.1 Caractérisation de texture.

Il existe une multitude d'outils qui permettent d'extraire certaines propriétés des textures et des images en général. On trouve d'une part les méthodes *spectrales* qui visent à décrire les évolutions du signal en utilisant le domaine fréquentiel. Les méthodes *statistiques* tentent elles de modéliser les textures pour extraire les paramètres déduits de ces modèles. Dans les deux cas, il faut souvent construire une description de l'image ou de la texture. Les outils utilisés sont appelés *descripteurs*, ils permettent de résumer une image en un nombre réduit de valeurs.

Une des qualités principales demandées aux descripteurs, dans l'optique de classifier et reconnaître des objets, réside dans leurs capacité à être invariant suivant plusieurs transformations telles qui la translation, la rotation et le facteur d'échelle par exemple.

Prenons le cas de la translation : si le but de l'outil est de reconnaître une voiture et que deux images sont testées avec la même voiture à deux positions différentes, il faut que le descripteur de l'image retourne la même information.

Quelques outils représentatifs de chaque méthode sont détaillés dans les deux paragraphes suivants.

3.1.1 Les méthodes statistiques.

Cette première famille de méthodes de description est fondée sur l'étude des propriétés statistiques, structurelles, ou des deux à la fois. Dans le cas des méthodes statistiques, on retrouve bien sûr les champs de Markov qui ont une place prépondérante dans les modèles choisis pour les algorithmes de synthèse de texture, décrits dans le chapitre 1. Les statistiques du premier ordre permettent d'extraire certaines propriétés du signal. Une méthode utilisant un modèle autorégressif sur plusieurs niveaux d'une pyramide Gaussienne est présentée dans [CD99]. Les étages de la pyramide contenant les étiquettes des régions sont assimilés à des champs Markoviens. Un algorithme itératif EM fait ensuite converger la segmentation et l'étiquetage des régions. Cependant, la méthode proposée permet de segmenter un nombre *connu* de régions (méthode *supervisée*), ce qui n'est pas le cas dans notre contexte d'étude.

La fonction d'autocorrélation AC est un des outils permettant de déterminer certains aspects des textures. Elle est définie pour une fenêtre carrée de taille $M \times M$ sur une image I par :

$$AC_{\Delta u,\Delta v}(I) = \frac{\sum_{u=0}^{M-1}\sum_{v=0}^{M-1} I(u,v)I(u+\Delta u, v+\Delta v)}{\sum_{u=0}^{M-1}\sum_{v=0}^{M-1} I^2(u,v)} \tag{3.1}$$

avec Δu et Δv les déplacements horizontaux et verticaux. Cette fonction permet notamment de juger de la finesse de la texture, mais surtout de la régularité des motifs. Plus les motifs seront réguliers, plus la fonction d'autocorrélation présentera des pics réguliers.

Une des méthodes les plus populaires pour la discrimination des textures est l'utilisation des matrices de cooccurrences qui mesurent la probabilité d'apparition des paires de valeurs de pixels suivant leur distance et direction. La dimension d'une matrice de cooccurrence dépend donc des valeurs prises par le signal soit 256×256 dans le cas de l'étude de la composante de luminance uniquement. Aussi, il est difficile de traiter directement l'information contenue dans cette matrice. C'est pourquoi il est nécessaire d'en extraire des paramètres dont les principaux ont été proposés par Haralick et al. [HSD73]. On peut citer pour les principaux d'en eux : l'énergie, la variance, le contraste, la corrélation ou l'entropie. Cette méthode permet de discriminer et peuvent ainsi permettre la segmentation. Dans notre optique de déterminer des paramètres tels que la taille des motifs, leur extraction depuis les informations de cooccurrence parait difficile.

Une dernière méthode utilisée pour la reconnaissance des textures s'appuie sur les LBP (Local Binary Patterns), introduits par T. Ojala dans [OPH96], qui permettent de décrire les voisinages des pixels en un vecteur. Un exemple fourni dans [TMT00], et illustré par la figure 3.1 permet de comprendre le processus de construction du descripteur pour un pixel, à partir de son voisinage 3×3. Une approche multirésolution est aussi proposée dans [TMT00]. Des deux méthodes possibles, à savoir la prise en compte de voisinages plus grands (5×5, 7×7...) et l'utilisation des voisinages 3×3 sur des niveaux sous-échantillonnés de la texture d'origine, les auteurs choisissent la première, qui permet d'éviter les contraintes de taille d'image et de filtre.

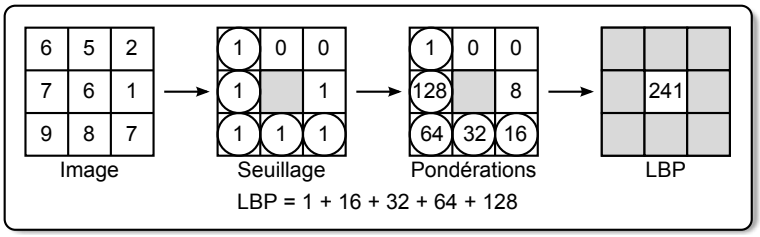


Figure 3.1 – *Construction du LPB pour un pixel à partir de son voisinage* 3×3.

De plus, la variation de la taille, forme, et orientation de ces voisinages permettent d'avoir des informations sur les motifs des textures. Il est à noter que les LBP sont, par essence, invariants par translation mais pas par rotation.

3.1.2 Les méthodes spectrales.

La transformée de Fourier (TF) est au départ l'outil fondamental pour les études harmoniques. Elle contient deux informations qui sont le *module* et la *phase*. Bien souvent, l'étude unique du module ne suffit pas lors de la description de larges blocs ou images. Les transformées discrètes telles que la DCT ou la transformée de Walsh/Hadamard peuvent alors apporter leur efficacité du fait qu'elles fournissent un signal purement réel alors que la TF est complexe. La DCT permet notamment une précision en fréquence deux fois plus grande que le module de la transformée de Fourier. La plupart des approches spectrales récentes ont néanmoins délaissé les transformées de Fourier et DCT, fondées sur des sinusoïdes infinies, avec l'avènement des ondelettes.

Une approche intéressante, présentée dans [SC94], couple segmentation et descripteurs en vue d'indexer les images en fonction des textures. Fondée sur une description orientée ondelettes, la méthode utilise une approche quadtree pour une première segmentation par blocs. La transformée en ondelette permet ensuite de caractériser les blocs résultants. Bien qu'elle utilise les ondelettes, les auteurs soulignent qu'une approche DCT peut très bien donner des résultats pertinents, l'approche étant ainsi aisément couplées aux systèmes standards de codage.

Les ondelettes semblent être une bonne approche de caractérisation, au regard de notre contexte d'étude. Un choix a cependant dû être opéré, afin de calculer rapidement des descripteurs nous permettant de continuer le reste du schéma proposé. Afin de rester cohérent avec ce dernier, une approche dans le domaine DCT a été choisie, qui permettra une réponse pertinente en fonction de la quantification opérée dans le même domaine. L'utilisation d'un codeur joint reposant sur l'utilisation d'ondelette pourra néanmoins conduire à la modification de cette méthode. Inspirée de descripteurs provenant du domaine de Fourier, cette dernière sera détaillée dans la section suivante.

3.2 Approche choisie : les descripteurs DCT.

Plusieurs informations paraissent importantes dans le contexte de la caractérisation pour la synthèse, à savoir :

- La texture est-elle détaillée / lissée ?
- La texture est-elle régulière périodique / stochastique ?
- Quelle est la taille des motifs ?

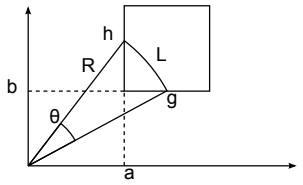


Figure 3.2 – *Calcul de l'intégrale sur une grille discrète de pixels.*

Il convient donc d'utiliser des outils discriminants pour ces caractéristiques en particulier. Pour rappel, dans le chapitre 1 est présentée une classification des textures provenant de [LLH04] qui ordonne les textures des plus régulières aux plus stochastiques. On peut penser au même genre de ≪ spectre ≫ de texture suivant le niveau de détails comme illustré par la figure 1.3 ou encore suivant la taille des motifs dont un classement est montré en figure 1.4.

Cette section décrit les descripteurs qui seront utilisés dans le schéma de codage présenté dans les chapitres 4 et 5. Avant cela, le cheminement vers la construction de ces descripteurs est détaillé, en commençant par les outils fournis par la littérature : les descripteurs de Fourier généralisés. Nous avons fait le choix d'une caractérisation spectrale afin de pouvoir extraire des périodes et ainsi des tailles de motifs qui vont permettre de fournir certains paramètres aux algorithmes de synthèse de texture.

3.2.1 Les descripteurs de Fourier généralisés.

L'utilisation des Descripteurs de Fourier Généralisés (DFG), issue des travaux de F. Smach, a été notamment présentée dans [SLG+08]. Ces travaux s'inscrivent dans le contexte de la classification d'objets. Le but affiché est donc de construire des descripteurs robustes, notamment en termes d'invariances afin de pouvoir reconnaitre des objets dans différents environnements et positions.

Mathématiquement, pour un signal continu, le calcul des DFG s'opère sur une fenêtre carrée, après avoir extrait la transformée de Fourier donnée par

$$F(\xi) = \int_{\mathbb{R}^2} f(x).e^{-j<x|\xi>}dx, \tag{3.2}$$

où f est sommable et $< .|. >$ est le produit scalaire sur $\mathbb{R}^2$.

Dans sa thèse [Sma09], F. Smach propose alors pour le calcul des DFG de passer en coordonnées polaires en modifiant le plan transformé afin que le coefficient de composante continue (DC) soit central. Ainsi si $F(\lambda, \theta)$ est la représentation en coordonnées polaires de la transformée, les descripteurs sont définis par

$$D_F(\lambda) = \int_{\theta=0}^{2\pi} |F(\lambda, \theta)|^2 \, d\theta. \tag{3.3}$$

La difficulté vient de l'application de ces descripteurs sur une grille discrète de pixels. Du fait que le schéma utilise une transformée discrète, il faut calculer la longueur de l'arc de cercle traversant chaque coefficient transformé, modélisé par un carré. La figure 3.2 illustre alors les paramètres à prendre en compte afin de calculer un coefficient particulier à la position (i, j). La valeur du coefficient transformé est pondérée par la longueur L

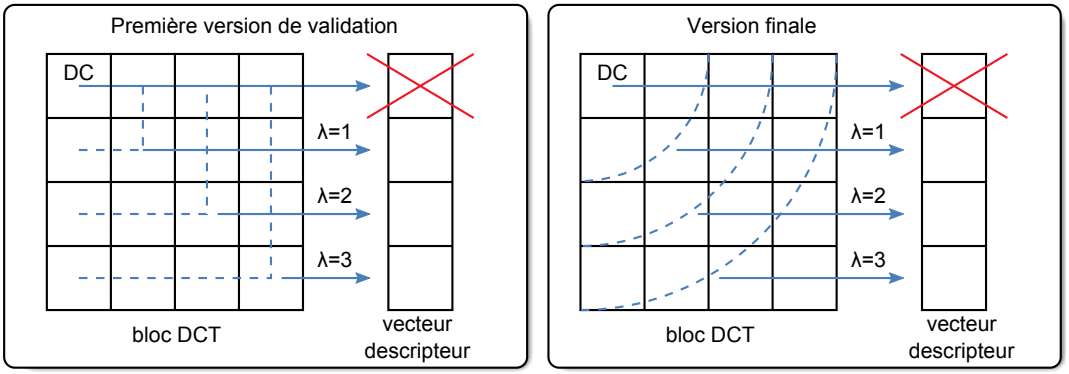


Figure 3.3 – *Illustration des chemins choisis pour le calcul des intégrales. Une première version simplifiée est présentée à gauche, la version finale à droite.*

de l'arc de cercle qui traverse le carré représentant le pixel en (i, j) de la position a à la position b. Cette longueur vaut

$$L = R * \theta \tag{3.4}$$

où

$$\theta = arcsin\left(\sqrt{1 - \frac{b^2}{R^2}}\right) - arcsin\left(\frac{a}{R}\right). \tag{3.5}$$

La transformée étant appliquée sur un signal réel, soit le signal $f(n)$ avec $n \in [0, N-1]$, on a :

$$F(k) = F^*(N - k). \tag{3.6}$$

La transformée discrète de Fourier est donc redondante et son spectre réel contient $\frac{N}{2} + 1$ éléments. Ainsi pour une image de 128×128 pixels par exemple, on obtient un vecteur de 64 composantes. Afin de produire des descripteurs invariants par variation globale d'éclairage, le vecteur est normalisé par la première composante. 63 coefficients permettent alors de décrire le signal.

Les DFG sont aussi invariants par déplacement. En effet, supposons $T(x, y)$ une fonction de translation. Soit $f(x, y)$ ayant pour transformée de Fourier $F(u, v)$, et $g(x, y) = f(x, y) \circ T(x, y) = f(x - x_0, y - y_0)$ ayant pour transformée $G(u, v)$. On a alors

$$G(u, v) = F(u, v).e^{-i2\pi(x_0+y_0)} \Rightarrow |G(u, v)| = |F(u, v)|, \tag{3.7}$$

d'où l'invariance.

Même s'ils ne sont pas invariants par changement d'échelle, l'auteur démontre cependant dans [Sma09] que si $g(x, y) = f(kx, ky)$, avec $k > 0$, alors

$$D_g(\lambda) = \frac{1}{k^4} D_f\left(\frac{\lambda}{k}\right). \tag{3.8}$$

Il est donc possible de détecter un facteur d'échelle à la lumière de cette propriété.

La faible résolution en fréquences spatiales de tels descripteurs nous a guidés vers l'utilisation de descripteurs utilisant le domaine DCT, qui sont décrits dans la section suivante.

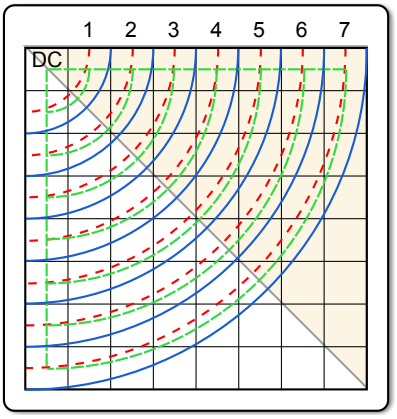


Figure 3.4 – *Intégrales possibles pour le calcul des descripteurs. Les arcs de cercle bleus en traits pleins ont été retenus.*

3.2.2 Création des descripteurs DCT.

Les descripteurs de Fourier ont été construits dans le but de reconnaitre des textures ou des objets dans des images typiquement de taille 128×128. Dans notre futur schéma de compression, il s'avère cependant que cette taille de fenêtre est prohibitive. En effet, il parait difficile ou du moins rare de pouvoir inscrire de telles fenêtres dans les régions segmentées. L'utilisation de fenêtres plus petites mais aussi les aspects pratiques nous ont alors orienté vers un outil basé transformée DCT. Contrairement au domaine de Fourier, la DCT transforme un signal réel en coefficients réels et permet surtout de transformer un signal discret de N composantes sur N coefficients. La résolution en fréquences spatiales est donc doublée, un avantage considérable pour créer un descripteur à partir de petits blocs. La DCT est utilisée en compression du fait du sa capacité à comprimer l'information sur les premiers coefficients par rapport à la transformée de Fourier. Ainsi, nos descripteurs seront plus robustes à la quantification des hautes fréquences.

De ce fait, il sera possible de découper l'image ou la région en blocs carrés de taille 4×4, 8×8, 16×16, 32×32... qui sont les tailles typiquement utilisées dans les standards de compression. Rappelons que les coefficients de la DCT 2D sur une fenêtre carrée de taille $N \times N$ sont calculés suivant

$$F(u,v) = \frac{1}{4} C_u C_v \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} I_{mn} \cos(u\pi \frac{2m+1}{2N}) \cos(v\pi \frac{2n+1}{2N}) \tag{3.9}$$

où

$$C_u = \begin{cases} \frac{1}{\sqrt{2}} & \text{si } u = 0 \\ 1 & \text{sinon} \end{cases} \quad \text{et } C_v = \begin{cases} \frac{1}{\sqrt{2}} & \text{si } v = 0 \\ 1 & \text{sinon} \end{cases} \tag{3.10}$$

Dans le but de valider certaines propriétés avant de résoudre le problème du calcul d'intégrales sur une grille discrète, une version simple a d'abord été implémentée. Les arcs de cercles utilisés pour les descripteurs de Fourier sont remplacés par les segments illustrés sur la figure 3.3. Les intégrales correspondent à la somme des coefficients traversés, elles permettent de valider une première version des descripteurs pour extraire une approximation de la taille des motifs.

Cependant, afin d'avoir des descripteurs fidèles aux fréquences spatiales qu'ils représentent, les intégrales circulaires sont nécessaires. Après le passage en coordonnées polaires comme pour les descripteurs de Fourier, les coefficients du vecteur descripteur DCT sont donnés par

$$D_{DCT}(\lambda) = \int_{\theta=0}^{\pi/2} |F(\lambda, \theta)|^2 \tag{3.11}$$

où $F(\lambda, \theta)$ correspond au coefficient DCT à la position $(\lambda \cos\theta, \lambda \sin\theta)$

Afin de calculer les intégrales le long de cercles concentriques censés représenter des « composantes » de fréquence spatiales, plusieurs arcs de cercles peuvent être pris en compte. La figure 3.4 répertorie trois variantes suivant quels rayons sont pris en compte et quel centre est choisi. En effet, dans le cadre de la transformée de Fourier 2D, centrée sur la composante continue comme la calcule F. Smach, le problème semble trivial grâce au centre. Dans le cas de la DCT, nous avons finalement opté pour les arcs de cercles illustrés en traits pleins. Une *Look Up Table* (LUT) répertoriant les longueurs permet d'accéder rapidement aux pondérations correspondantes.

Avant de décrire leur utilisation, quelques tests de robustesse aux transformations ont été réalisés.

3.2.3 Tests sur les invariances.

Les invariances des descripteurs sont des propriétés très recherchées dans l'optique de la reconnaissance d'objets. Cependant, dans le cadre de la caractérisation dans le but et de paramétrer nos algorithmes de synthèse, certaines invariances ne sont pas fondamentales, voire indésirables. En effet, dans le cas de l'utilisation des descripteurs dans la segmentation, il n'est pas souhaitable de segmenter toute une région dont les motifs varient en rotation si la synthèse n'est pas capable de gérer cette variation. La suite de cette section détaille les différents tests menés sur les invariances et leurs implications.

Invariance par changement de luminance.

Le fait de retirer la composante continue du descripteur permet une invariance par variation d'éclairage global. Ce type d'invariance est souhaité et relativement important car très souvent, la luminance va changer sur une même texture à cause d'une zone plus ombragée par exemple, alors qu'il s'agit de la même texture, avec les mêmes caractéristiques. L'adaptation naturelle des algorithmes permet de plus de gérer les variations de luminance.

Invariance par translation.

Les descripteurs DCT ne sont pas invariants par translation. Prenons l'exemple d'une translation suivant un axe Ox. Een reprenant les notations de la section 3.2.1, on a

$$\begin{array}{c} g(x) = f(x - x_0) \Rightarrow G(u) = \int_{-\infty}^{+\infty} f(x - x_0)\cos(2\pi x u)dx, \\ X = x - x_0 \Rightarrow G(u) = \int_{-\infty}^{+\infty} f(X)\cos(2\pi u(X + x_0))dx \neq F(u). \end{array} \tag{3.12}$$

Cette propriété est aisément vérifiable en calculant les descripteurs sur un signal synthétique que l'on décale.

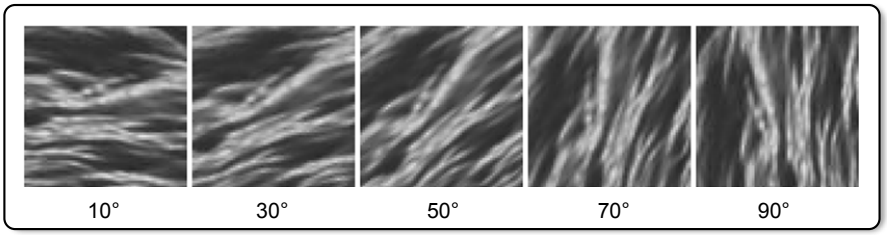


Figure 3.5 – *Rotation d'une texture de* River *en utilisant un filtre d'interpolation bilinéaire*

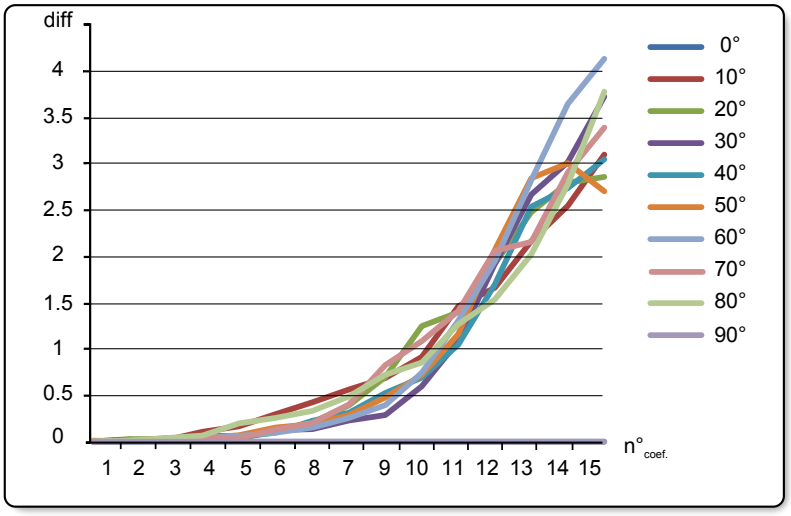


Figure 3.6 – *Évolution des descripteurs en fonction de l'angle de rotation.*

Invariance par rotation.

L'invariance par rotation a également été testée. Les différentes textures ont été tournées avec plusieurs valeurs d'angles. La difficulté du test réside ici dans le choix de l'interpolation choisie pour les valeurs d'angle différentes de 0° et 90°. En effet, plusieurs choix s'offraient dont la recherche du pixel voisin le plus proche, l'interpolation simple et l'interpolation bilinéaire. La première et la dernière ont été testées afin de détecter leurs effets sur les descripteurs. La figure 3.5 illustre l'exemple d'une texture représentant la surface d'une rivière avec une interpolation bilinéaire pour quelques angles. Les différences entre les vecteurs moyens des textures tournées par rapport au vecteur moyen de la texture originale sont présentées sur la figure 3.6. Il apparaît clairement que les descripteurs ne sont pas invariants en rotation.

Invariance par quantification.

Dans le domaine de la compression, il est intéressant d'avoir les descripteurs les plus robustes possibles aux effets de la quantification. Nous avons donc calculés les descripteurs de la même zone mais quantifiée à des taux différents, en utilisant les matrices de quantification présentes dans le standard JPEG. La figure 3.7 présente deux niveaux de quantification d'un patch de texture représentant de l'osier. On constate alors sur la figure 3.8 que les descripteurs sont différents d'un taux de quantification à l'autre. Cependant,

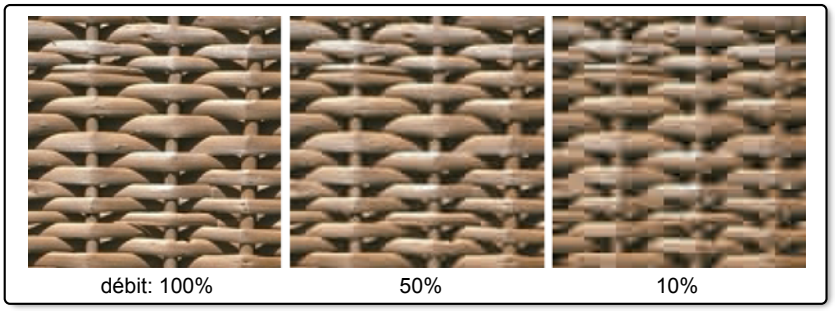


Figure 3.7 – *Quantification de la texture d'osier* fabric 14 *de la base* Vistex

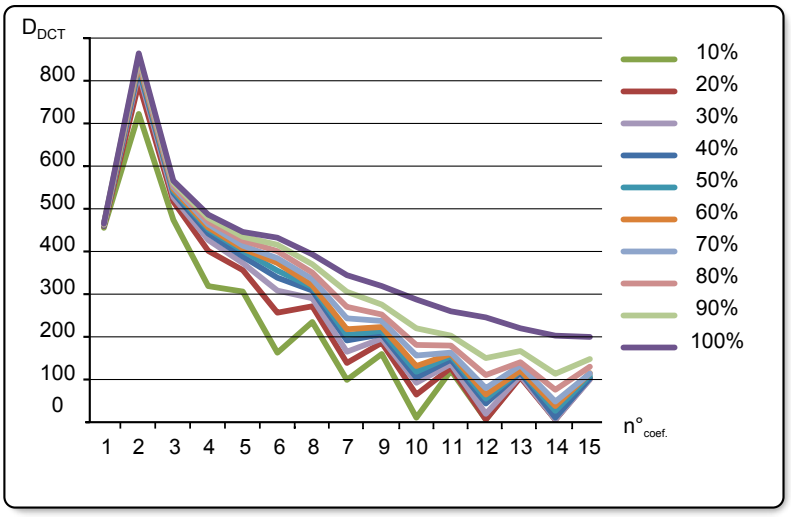


Figure 3.8 – *Courbes des descripteurs moyens obtenus sur l'image* Osier 256 × 256 *suivant le taux de quantification de type JPEG.*

on note que les descripteurs sont sensibles sur les hautes fréquences qui ont été sacrifiées par la quantification : ils gardent cependant la même tendance du moins pour les basses fréquences. Il est donc toujours possible d'extraire certaines caractéristiques de ces vecteurs.

La section suivante présente la première utilisation des descripteurs DCT à l'intérieur du schéma de codage, à savoir l'envoi d'informations permettant les décisions des paramètres de synthèse.

Conséquence sur l'utilisation des descripteurs.

Les descripteurs n'étant pas invariants par translation, ils n'en sont pas moins légitimes pour détecter certaines caractéristiques des textures présentées ci-après. Il a donc été décidé de procéder au calcul d'un grand nombre de descripteurs pris à des positions aléatoires sur la région considérée. Un vecteur moyen peut ensuite être calculé afin de représenter la région. Cette utilisation sera reprise dans la suite de ce manuscrit.

La section suivante présente l'utilisation première des descripteurs DCT dans notre schéma, soit la détermination de paramètres pour les algorithmes de synthèse de texture.

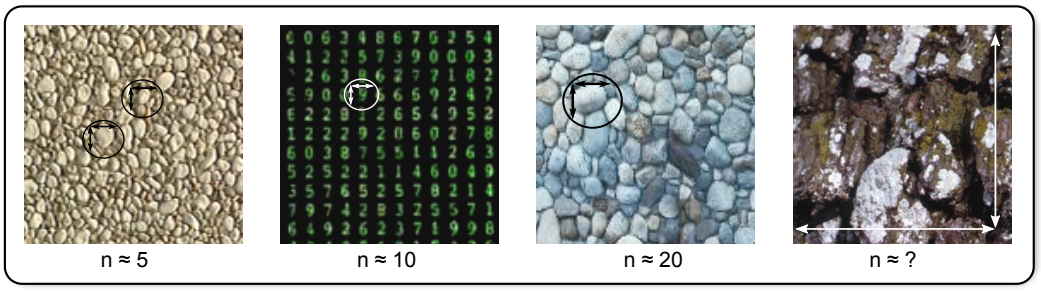


Figure 3.9 – *Tailles caractéristiques des motifs.*

3.2.4 Détection de paramètres.

Les algorithmes de synthèse de texture utilisés, à savoir des méthodes orientées *pixel* et *patch*, ont pour paramètre principal une taille de voisinage (resp. patch) optimale, suivant le type de motif. Il a été présenté dans le chapitre 1 que pour obtenir des résultats acceptables de synthèse, ce paramètre de taille devait être supérieur à la taille d'un motif élémentaire de la texture. Le terme de motif élémentaire a un sens très large étant donné l'aspect stochastique de la plupart des textures. Les échantillons présentés en figure 3.9 démontrent que dans certain cas, il est assez aisé de déterminer une taille en dessous de laquelle une fenêtre serait trop petite pour contenir un motif (*i.e.* les trois textures de gauche). Dans d'autres cas comme pour la texture de droite, cette taille sera maximum.

Un premier critère non retenu.

Il existe par construction un rapport 2 entre la composante d'un vecteur calculé sur un bloc de taille N et le coefficient correspondant dans le vecteur calculé sur un bloc de taille double, soit

$$\frac{D_N(i)}{D_{2N}(2i)} = 2, \forall i \in]0, N], \tag{3.13}$$

avec N la taille du vecteur. Pour trouver la taille d'un patch décrivant correctement la texture, ou pour trouver un voisinage suffisant pour caractériser la texture en synthèse de texture, un processus envisageable serait de calculer les descripteurs sur des tailles de bloc de plus en plus petites (tant que le rapport de 2 est vérifié) avec une tolérance donnée. Cette tolérance passée, le nouveau bloc DCT sera considéré comme trop petit pour contenir un motif de texture.

Cette méthode a été testée sur 6 patchs présentés en figure 3.10. Le tableau 3.1 présente les rapports moyens obtenus. Ces résultats proviennent en fait d'une moyenne à deux niveaux.

- Comme énoncé précédemment, un grand nombre de descripteurs ont été calculés sur la surface des patchs à des positions aléatoires. Un descripteur de chaque taille est ainsi issu d'une fenêtre centrée sur chaque position. Pour chaque taille, un vecteur moyen est enfin calculé et comparé.
- Le rapport recherché est calculé entre 2 composantes prises sur 2 descripteurs moyens de taille différente.

  Ainsi, par exemple pour deux descripteurs D_4 et D_8, calculés sur des blocs de tailles respectives 4×4 et 8×8, on regarde les rapports $\frac{D_4(1)}{D_8(2)}$, $\frac{D_4(2)}{D_8(4)}$, $\frac{D_4(3)}{D_8(6)}$, $\frac{D_4(4)}{D_8(8)}$. Le tableau

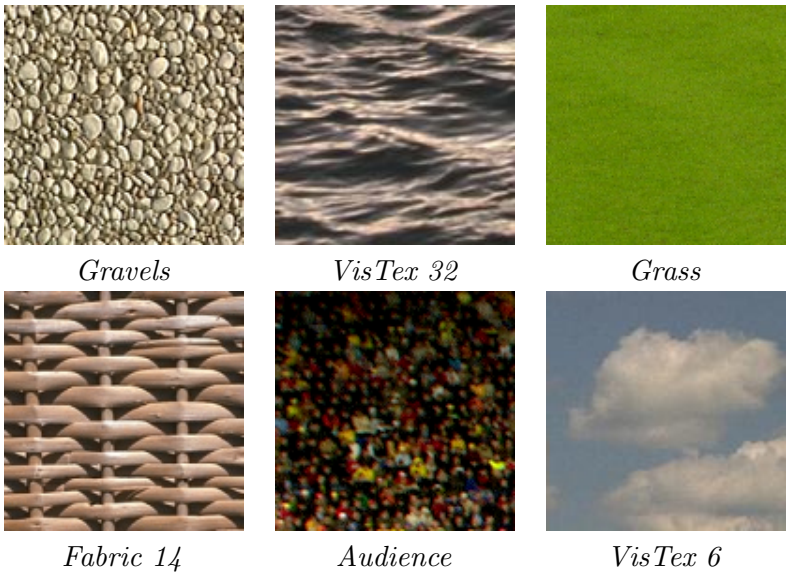


Figure 3.10 – *Patchs de texture utilisés pour les tests des descripteurs DCT. Les patchs numérotés proviennent de la base* VisTex *de textures [PGM+95], les autres ont été extraits dans des images.*

	Gravels	*VisTex 32*	*Grass*	*Fabric 14*	*Audience*	*VisTex 6*
4×4 / 8×8	1,84	1,90	2,07	2,27	1,81	2,01
8×8 / 16×16	1,87	1,99	2,32	1,94	1,91	1,95
16×16 / 32×32	1,93	1,95	2,30	1,84	1,94	1,96

Table 3.1 – *Moyennes des rapports entres les coefficients DCT suivant l'équation 3.13.*

3.1 présente la moyenne de ces N rapports pour la comparaison des descripteurs provenant de blocs DCT de tailles $N \times N$ et $2N \times 2N$.

On s'aperçoit malheureusement que si pour certaines textures comme *Fabric 14*, le rapport 2 est assez nettement perdu en dessous d'un bloc de 16×16, il faudrait pour la plupart des textures déterminer des seuils complexes. Une méthode serait de faire un apprentissage sur une grande base de données de textures, puis de recouper avec les résultats des synthèses pour chaque taille de fenêtre, afin de trouver des paramètres de seuillage pertinents. Cette méthode a donc été abandonnée au profit de celle présentée dans le paragraphe suivant, qui présente l'avantage de ne pas introduire de seuil.

Variations des coefficients du descripteur.

La méthode retenue pour notre schéma tient compte des variations de signal représentées par les coefficients traversés. La figure 3.11 présente les variations de signal correspondants aux atomes DCT entrant en compte dans le calcul des descripteurs. On s'aperçoit alors que le premier coefficient du vecteur descripteur contient des atomes qui correspondent à une variation unique du signal sur la fenêtre de taille N dans chaque direction. Ainsi, si ce signal est prépondérant et que la taille de fenêtre englobe un motif, il s'agirait de variations du type de celle présentée pour $D(1)$ à droite. Ce type de périodicité étant quasi improbable, il semble que la taille de fenêtre soit trop petite par rapport au motif.

Au contraire, les coefficients suivant font entrer en jeu des coefficients pour lesquels le

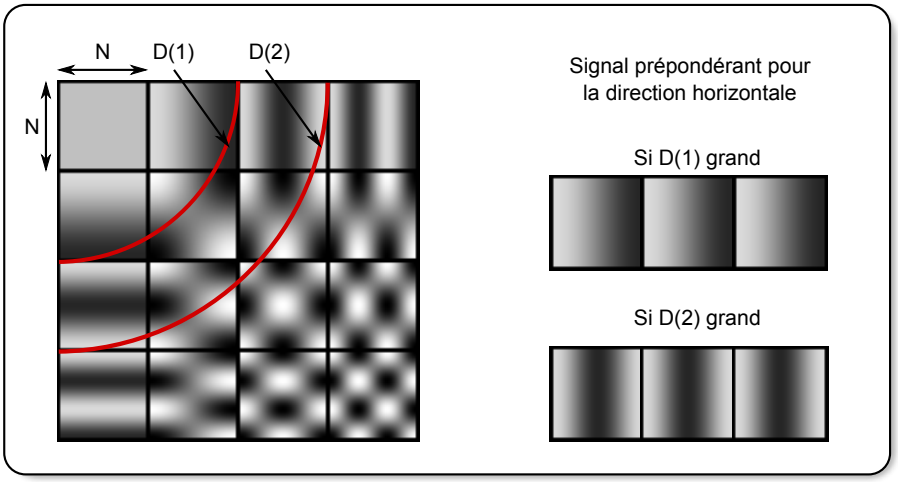


Figure 3.11 – *Atomes DCT et descripteurs.*

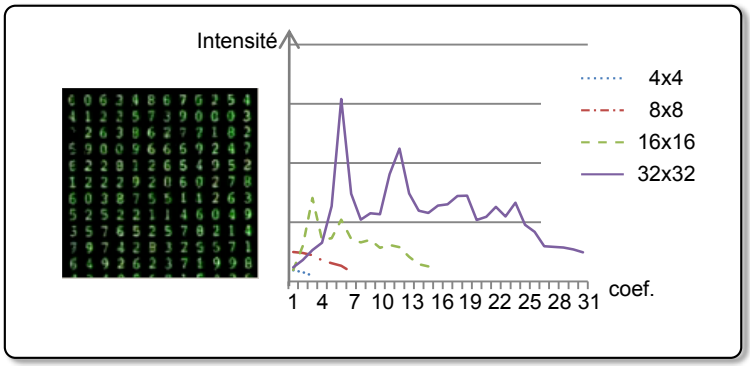


Figure 3.12 – *Tracé d'un vecteur descripteur pour plusieurs tailles de fenêtre.*

signal observe au moins une période, comme présenté en figure 3.11 pour $D(2)$. Si ce type de variation est prépondérant, nous faisons l'hypothèse que le signal texturé observe assez de variations à l'intérieur de la fenêtre sur laquelle est calculé le bloc DCT.

L'opération de sélection de la taille de fenêtre adéquate nécessite donc de tracer les vecteurs descripteurs suivant l'ordre des coefficients, comme illustré en figure 3.12 et 3.13.

Si la courbe obtenue est globalement décroissante, *i.e.* elle a pour maximum le premier coefficient, alors c'est que la taille de fenêtre n'est pas assez grande pour englober un motif complet. Ce critère de monotonie permet ainsi de déterminer automatiquement la taille des motifs de nos textures. Il a pour avantage d'éviter de fixer des seuils puisque seule les valeurs relatives des coefficients est prise en compte, contrairement à la méthode précédente, ainsi que pour la plupart des méthodes spectrales.

Aussi, d'après la Figure 3.12, la texture *Matrix* aura besoin d'une taille de fenêtre de 16, qui permet d'englober le motif (*i.e.* un caractère estimé à 10×10 pixels). Cette méthode a été ensuite validée sur 20 patchs de texture dont 6 caractéristiques sont présentés en figure 3.13. On note que pour les nuages de la texture *VisTex 6*, aucune taille ne convient puisqu'il est impossible de détacher un motif si ce n'est le patch entier. Empiriquement, la synthèse de tels patchs avec la méthode de Wei donne de bons résultats avec des voisinages

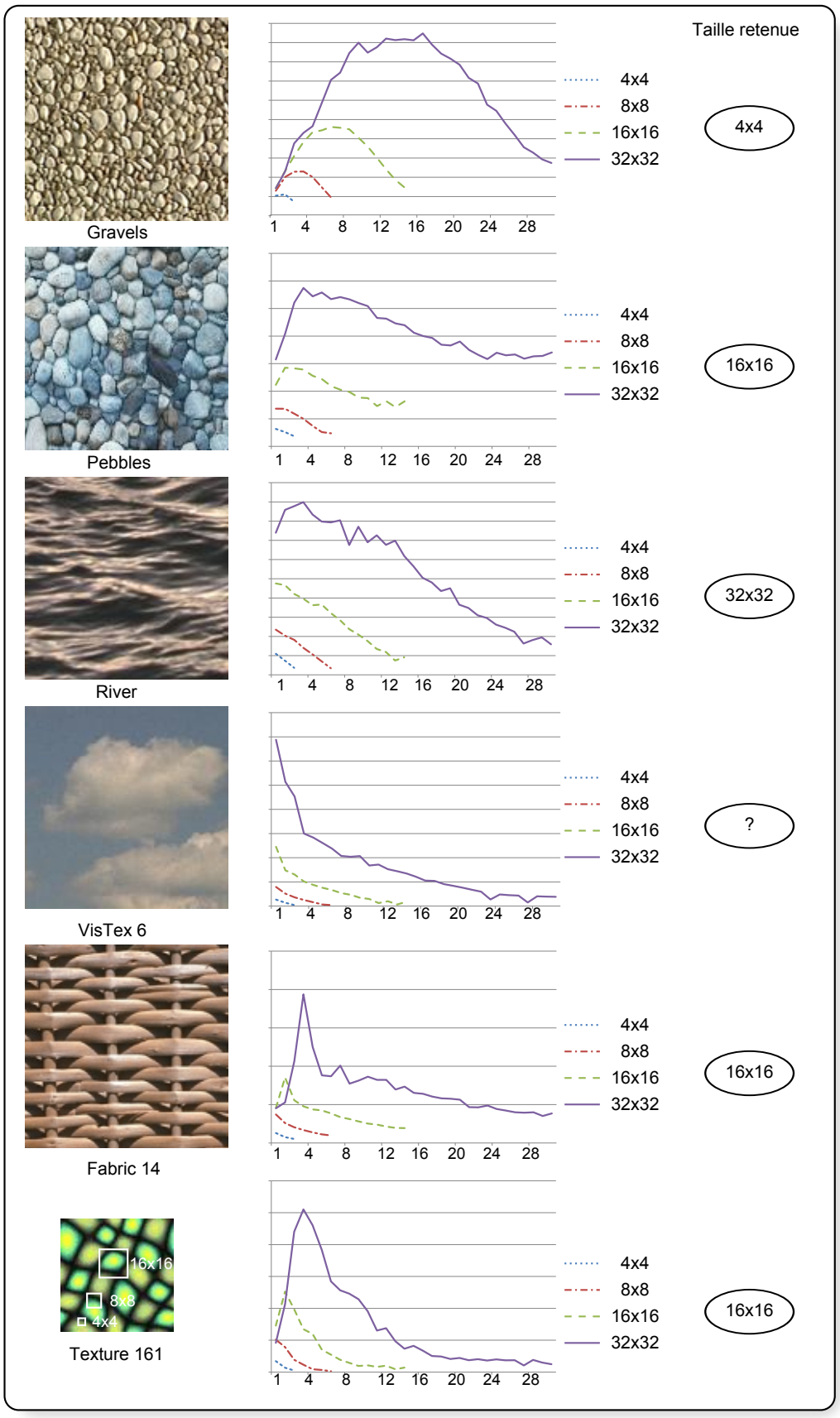


Figure 3.13 – *Tracé des vecteurs descripteurs pour plusieurs patchs de texture.*

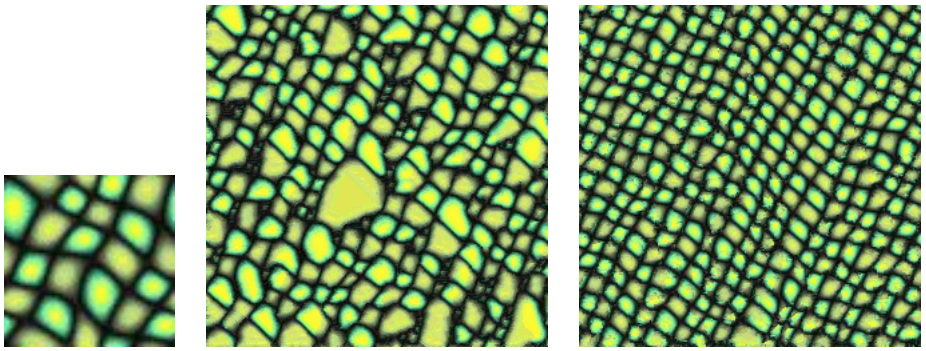


Figure 3.14 – *Résultats de la synthèse de L.Y. Wei [WL00] suivant la taille de voisinage utilisée.*

de taille supérieure à 8×8. On verra dans le chapitre suivant comment traiter ce cas suivant les algorithmes de synthèse choisis. La figure 3.14 montre que la qualité de la synthèse en utilisant un voisinage 7×7 ne permet par un résultat satisfaisant (des tailles impaires sont choisies pour des raisons de commodité avec l'algorithme de L.Y Wei). Avec un voisinage 15×15, proche de la taille de descripteur choisie, la synthèse est visuellement satisfaisante.

Le paramètre de taille étant déterminé via l'utilisation des descripteurs DCT, les sections suivantes présentent maintenant les outils de segmentation, essentiels aux schémas de codage orientés synthèse de texture. Nous verrons ensuite que ces descripteurs ont aussi une utilité pour discriminer les régions texturées, en cohérence avec la caractérisation.

3.3 Travaux de segmentation.

La segmentation vise à partitionner le domaine spatial d'une image en sous-ensembles non chevauchés : les régions. Ces dernières sont homogènes par rapport à une certaine propriété telle que la teinte ou la texture. Cette propriété diffère sensiblement entre deux régions adjacentes afin de pouvoir les distinguer. Enfin, l'union de ces régions donne l'image complète [HS85]. Il est difficile voire impossible, dans le domaine de la vision artificielle, de reproduire une segmentation parfaite, c'est à dire équivalente à celle qu'un humain dessinerait sur l'image. En effet, la perception humaine se fait sur deux niveaux :

- la segmentation *bas niveau* qui découpe des régions suivant leur homogénéité sans prendre en compte la reconnaissance des objets dans la scène. C'est celle que l'on va tenter d'approcher.
- la segmentation *haut niveau* qui prend en compte la *sémantique*, *i.e.* l'humain extrait les objets qui ont un sens pour lui [EK98]. Ces objets peuvent être constitués d'une ou plusieurs régions.

La segmentation des images est une discipline essentielle et très répandue dans le traitement des images. Malheureusement, il n'existe pas de solution idéale permettant de segmenter automatiquement des scènes aussi bien qu'un individu. Il faut donc trouver une solution adaptée aux circonstances, *i.e* la détection d'objets, de régions cohérentes, de premiers plans... Pour la segmentation de régions texturées, au sens défini dans le premier chapitre, plusieurs approches peuvent être envisagées, que l'on regrouper en trois grandes catégories.

- La segmentation fondée sur la détection de **contours**. Cette approche consiste à localiser les frontières présentes autours des objets ou des régions en général de

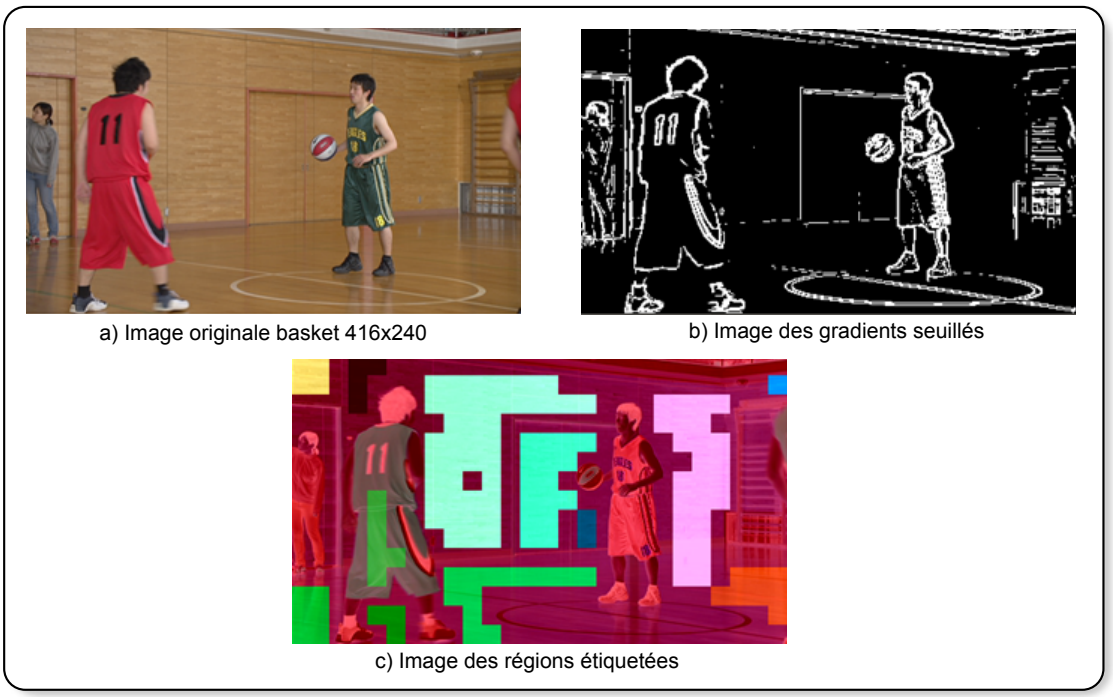


Figure 3.15 – *Segmentation d'une image de la séquence* Basket *via l'extraction des gradients et l'étiquetage des régions sur une grille* 16×16.

l'image. Il convient ensuite *d'étiqueter* chaque région ainsi détourée pour obtenir une segmentation de l'image.

- La segmentation fondée sur la **croissance de régions** (*region growing*). Il s'agit pour cette approche de partir de graines données ou aléatoires pour les faire grandir par fusion suivant un critère, jusqu'au remplissage total de l'image.
- La segmentation fondée sur la **fusion de régions** (*region merging*) qui consiste d'abord à diviser (*split*) l'image à l'aide d'un critère d'homogénéité, puis de fusionner (*merge*) les zones adjacentes pour créer des régions cohérentes.

Certaines techniques dites coopératives utilisent plusieurs approches pour tenter d'exploiter les avantages des deux méthodes. Enfin, pour le traitement des vidéos, une segmentation utilisant les données liées au mouvement peut être envisagée seule ou couplée aux segmentations spatiales.

Au départ, l'idée était d'utiliser des outils simples afin de poursuivre l'optimisation des étapes de raffinement et de synthèse. Une approche simple, par détection de contour a donc, d'abord, été implémentée.

3.3.1 Détection de contours.

Cette méthode nécessite de trouver les pixels de *contours*, c'est à dire les fortes variations locales de couleur ou d'intensité lumineuse.

Plusieurs méthodes peuvent être envisagées pour cette détection.

- Une méthode par seuillage sur les niveaux de gris ou sur des informations locales comme les matrices de cooccurrence.
- L'analyse de l'histogramme des luminances afin de détecter les différentes classes de pixels.

- L'extension de la méthode précédente à l'étude conjointe des histogrammes d'intensité lumineuse et des couleurs.
- L'analyse d'un vecteur gradient calculé sur la composante des luminances.
- L'analyse d'un vecteur gradient résultant de gradients calculés pour chaque composante pour les images en couleurs.

Les trois premières méthodes ont l'inconvénient de ne prendre en compte que les histogrammes et non les informations de localisation spatiales, ce qui les rend vulnérables aux bruits. Cependant, dans le cas d'images avec des régions bien distinctes au niveau de l'intensité lumineuse, l'étude des histogrammes correspondants peut permettre une délimitation efficace. En effet, ils font clairement apparaître des pics avec des vallées qui les séparent nettement. Il est alors possible de délimiter les régions en cherchant les minima locaux de l'histogramme étudié. Dans [SSW88] sont décrites des techniques plus robustes aux bruits et problèmes spatiaux, et ont donc été mises en places, mais restent cependant perfectibles. Malheureusement, dans le cas notamment des vidéos naturelles, les régions texturées à délimiter peuvent contenir des motifs animés d'une grande dynamique. Il est ainsi impossible de définir des seuils permettant de séparer efficacement des régions cohérentes visuellement.

Les gradients étant obtenus par des calculs de dérivées, ces méthodes sont aussi appelées dérivatives. Plusieurs filtres peuvent ainsi être utilisés dont les plus répandus : filtres de Sobel, Kirsh, Prewitt... Même si les méthodes utilisant les gradients nécessitent aussi de forts contrastes dans les régions à délimiter, ils paraissent fournir un premier outil simple d'utilisation afin de continuer d'optimiser les étapes fondamentales de synthèse de texture.

Cependant, il n'est pas évident de construire des contours fermées à partir de ces méthodes, permettant de délimiter les régions. En effet, la détection de discontinuités n'aboutit pas forcément à une segmentation puisque les contours ne créent pas forcément des domaines connexes.

La figure 3.15 illustre l'exemple d'un premier outil orienté contours, implémenté afin de pouvoir se focaliser sur les autres étapes des schémas proposés dans ces travaux de thèse. Des filtres de Sobel sont utilisés afin de détecter les contours de l'image, dont les noyaux horizontaux et verticaux sont respectivement donnés sur une image $\mathbf{I}$ par

$$\mathbf{G_h} = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{I} \text{ et } \mathbf{G_v} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{I} \tag{3.14}$$

et les noyaux quadratiques par

$$\mathbf{G_{q_1}} = \sqrt{2} \begin{bmatrix} +4 & +1 & 0 \\ +1 & 0 & -1 \\ 0 & -1 & -4 \end{bmatrix} * \mathbf{I} \text{ et } \mathbf{G_{q_2}} = \sqrt{2} \begin{bmatrix} 0 & -1 & -4 \\ +1 & 0 & -1 \\ +4 & +1 & 0 \end{bmatrix} * \mathbf{I}. \tag{3.15}$$

La valeur retenue du gradient total G_T pour chaque pixel s'écrit alors

$$G_T = \sqrt{G_h^2 + G_v^2 + G_{q_1}^2 + G_{q_2}^2} \tag{3.16}$$

Une méthode de seuillage permet ensuite d'obtenir l'image 3.15 b) qui vise à dresser la carte des contours présents. Une adaptation sur une grille 16×16 ainsi qu'un étiquetage des régions permettent enfin d'obtenir la répartition illustrée par l'image 3.15 c). La grille 16×16 a été adoptée pour des raisons de commodité d'utilisation de régions compatibles

avec le fonctionnement de l'encodage des MB du standard H.264. Les différentes couleurs correspondent aux régions segmentées.

Cette manipulation confirme la difficulté d'obtenir des contours fermés et ainsi des régions cohérentes via une méthode uniquement fondée sur la détection de forts gradients. La section suivante présente alors la méthode par croissance des régions permettant de construire une segmentation cohérente avec les textures étudiées.

3.3.2 Croissance de régions.

Afin de segmenter les régions en fonction de leur contenu, il faut pouvoir définir les propriétés que les pixels d'une même région ont en commun. L. Macaire dans [Mac04] définit une distribution en régions $\{R_i\}$ à partir des 4 critères qui doivent nécessairement être valides :

- $I = \cup_i R_i$,
- $\forall i$ les R_i sont constituées de pixels connexes,
- $Crit(R_i) = vrai, \forall i$,
- $Crit(R_i \cup R_j) = faux$, pour $i \neq j$ et R_i, R_j sont adjacentes.

$Crit(R_i)$ correspond ici à un critère d'uniformité qui doit être valide sur chaque région construite. Une image correctement segmentée doit ainsi respecter les 4 conditions.

Ainsi tous les pixels qui appartiennent à une région constituant un ensemble de pixels spatialement connexes et vérifient une contrainte d'homogénéité fixée. Il est à noter aussi qu'avec le même critère d'homogénéité pilotant le prédicat, il existe plusieurs solutions.

Les paragraphes suivants développent l'approche utilisée et son adaptation au contexte du codage. Comme pour la première version développée, le choix de l'outil tient compte les performances liées au contexte mais aussi l'aspect pratique d'utilisation d'une méthode prête et disponible. Cet outil provient en effet du codec LAR (Locally Adaptive Resolution), développé par le groupe image de l'IETR, qui propose parmi ses fonctionnalités la représentation des images en régions.

3.4 Outil de segmentation proposé.

3.4.1 La méthode LAR (Locally Adaptive Resolution).

Cette méthode conjointe de codage et de représentation est née de l'idée que la taille des blocs considérés pour le codage d'images devrait s'adapter à l'activité locale du signal. Si la luminance s'avère localement uniforme, une grande taille de bloc peut être efficace. A l'inverse, si l'activité reste localement élevée, une résolution importante parait nécessaire pour restituer l'activité locale. Dès lors, O. Déforges dans [Déf04] décrit un *codec* complet décomposant le codage basse résolution de celui des textures comme illustré sur la figure 4.1. La fonctionnalité du LAR qui nous intéresse dans cette thèse s'appuie sur la représentation spatiale des image, fondée sur une décomposition en *quadtree*. La suite s'attache donc à décrire succinctement son fonctionnement afin de pouvoir présenter l'approche région utilisée.

Ainsi, une grille de blocs s'apparentant à une collection de quadtrees est ainsi définie par la partition de l'image, comme illustré sur l'image (b) de la figure 3.17. L'image (c) présente le quadtree rempli des valeurs moyennes par bloc et l'image (d) l'image finale obtenue par la méthode LAR. Cette partition répond à un critère d'homogénéité qui traduit les propriétés d'activité locale du signal. Ce critère repose sur un *gradient morphologique*,

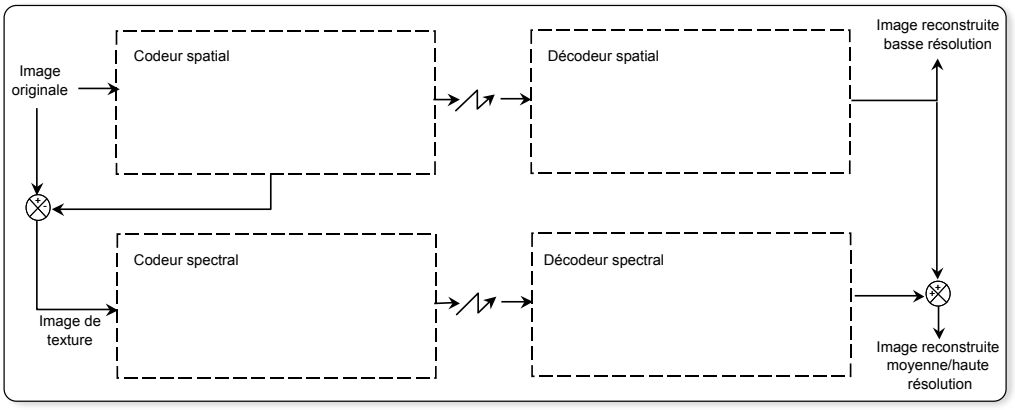


Figure 3.16 – *Schéma global LAR à deux couches : codeurs spatial et spectral*

i.e. la différence entre la valeur maximal et la valeur minimale du bloc considéré. Ainsi, le gradient doit être inférieur à un seuil sur le bloc considéré, mais supérieur au même seuil en considérant les blocs voisins. La méthode est récursive, initialisée par l'image des blocs de taille 2×2, ayant pour valeur les moyennes des blocs. Les fusions sont ensuite opérées jusqu'à une taille de bloc limite, définie par l'utilisateur. L'image obtenue à l'issue de cette étape est donc constituée de blocs de tailles variables représentés par leur valeur moyenne. Dans le cas des images couleur, la solution adoptée consiste à définir une seule partition régulière pour l'ensemble des trois composantes Y : Cr : Cb.

Le paragraphe suivant détaille l'application de représentation en régions du LAR.

3.4.2 LAR : approche région.

Cette fonctionnalité vise au départ à segmenter au codeur et au décodeur les images à partir d'une image bas débit, ceci afin d'éviter la transmission de la carte des régions. Elle constitue pour notre étude un outil de segmentation en régions construit à des fins de codage et compression, donc en accord avec l'utilisation souhaitée.

Afin de partitionner l'image en régions à partir de la structure du LAR spatial, il reste à fusionner et regrouper les blocs qui se ressemblent. Cette opération est cependant délicate, particulièrement pour la définition des critères de fusion, mais aussi pour la définition de l'ordre dans lequel ces opérations sont faites.

Soient $S = \{(x, y) | 1 \leq x \leq N_x, 1 \leq y \leq N_y\}$, les coordonnées spatiales des pixels dans une image de N_x lignes et N_y colonnes. Notons R^K l'ensemble des régions dans la partition Δ^K. La segmentation d'une image en K régions $R_k^K \in \Delta^K$ consiste à trouver la partition Δ^K de S telle que :

$$S = \Delta^K = \bigcup_{k=1}^{K} R_k^K, \tag{3.17}$$

avec $R_i^K \cap R_j^K = \emptyset$, $\forall (i, j) \in \{1 \dots K\}^2$ pour $i \neq j$.

Initialisé par une partition Δ^{K_0} ($K_0 \leq N_x \times N_y$), le but du procédé de segmentation est de transformer Δ^{K_0} en une nouvelle partition Δ^K ($K < K_0$) selon un critère d'homogénéité, et cela à travers des séquences de fusions de régions. Dans le cas présent, Δ^{K_0} correspond aux blocs de luminance issus du codeur spatial.

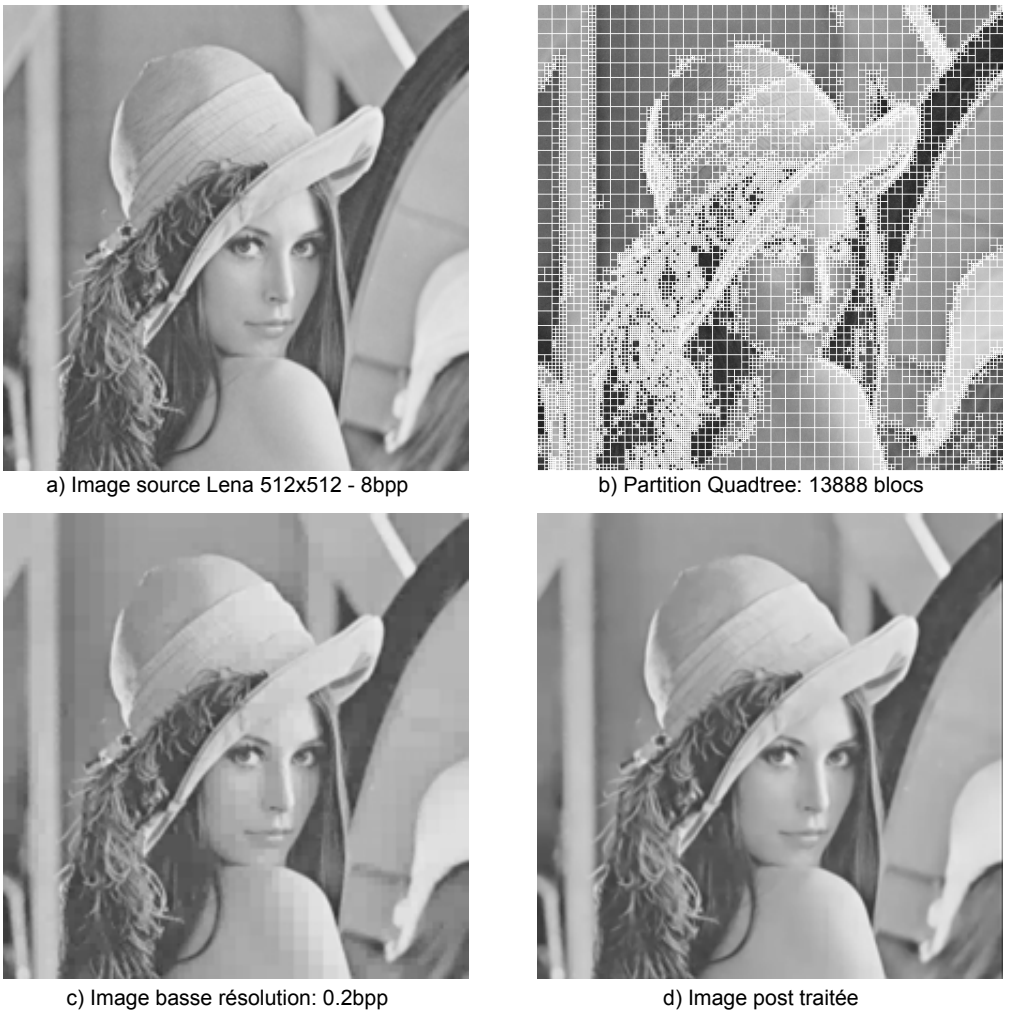


Figure 3.17 – *Résultas du LAR sur l'image* Lena *de taille* 512×512 *avec l'approche spatiale.*

Graphe d'adjacence

De manière naturelle, les régions reconstruites doivent former des ensembles spatialement connexes. Aussi, la relation d'adjacence est-elle au cœur des principes de segmentation. Nous appellerons par la suite A_i^K l'ensemble des régions connexes à $R_i^K \in \Delta^K$. Il faut donc, pour décider de la fusion ou non, mesurer la distance entre deux classes $Cost\left(R_i^K, R_j^K\right)$

La structure de données classique pour représenter des partitions est le ≪ Region Adjacency Graph ≫ (RAG) [SHB93]. Le RAG^K d'une K-partition est défini comme un graphe non orienté, $G^K = (V^K, E^K)$, où $V^K = \{1, \ldots K\}$ est l'ensemble des sommets et $E^K \subset V^K \times V^K$ est l'ensemble des arêtes. Chaque région est représentée par un sommet du graphe, et entre deux sommets (régions) $\left\{R_i^K,\ R_j^K\right\} \in V^K \times V^K$ il existe une arête (i, j) si les régions sont adjacentes.

Algorithme 3.1: Algorithme de fusion des régions

Entrées : K_0 : nombre de blocs de la partition initiale Δ^{K_0}.
Sorties : K : partition finale

$K = K_0$;
$Nb_{fusions} = 0$
tant que $Nb_{fusions_prec} < Nb_{fusions}$ **faire**
  $Nb_{fusions_prec} = Nb_{fusions}$;
  i=1;
  tant que $i \leq K_0$ **faire**
    si $R_i^K \in RAG^K$ **alors**
      Trouver $R_i^K \in A_i^K$
      tel que $Cost\left(R_i^K, R_j^K\right) \leq Cost\left(R_i^K, R_l^K\right), \forall R_l^K \in A_i^K$;
    fin
    Incrémenter i;
  fin
  i=1;
  tant que $i \leq K_0$ **faire**
    si $R_i^K \in RAG^K$ **alors**
      si $Cost'\left(R_i^K, R_j^K\right) < Th_{Cost}$ **alors**
        Fusionner R_i^K et R_j^K;
        Décrémenter K;
        Incrémenter $Nb_{fusions}$;
      fin
    fin
    Incrémenter i;
  fin
fin

Classification hiérarchique et métrique.

Fusionner les régions selon un critère d'homogénéité se ramène en général à un problème de classification hiérarchique, consistant à chercher les éléments les plus proches au sens d'une distance D, puis à mesurer les agrégations entre les classes suivant un critère $Crit$ donné. La hiérarchie est dite *indicée* si pour toute partie H de la hiérarchie, la relation d'inclusion $H \subset H' \Rightarrow D(H) \leq D(H')$. Un niveau hiérarchique donné correspond alors à la fusion entre un sommet et un ensemble de sommets connexes.

L'intérêt d'une approche par fusion selon la distance minimale réside dans le fait que l'on puisse exactement contrôler le nombre final de régions. Par construction, les régions fusionnées sont également les plus proches du point de vue de la distance choisie. Cet avantage n'est toutefois que relatif, car le nombre nécessaire de régions pour décrire « correctement » une image reste bien sûr dépendant de la complexité de celle-ci.

La méthode adoptée est fondée sur une classification hiérarchique qui considère trois seuils consécutivement. Pour cette application, seul le plus haut niveau de hiérarchie est utilisé. Cependant, la segmentation du premier niveau qui aboutit à des régions très homogènes, suivie des relâches successives sur les seuils de fusion, aboutit globalement sur des résultats meilleurs qu'avec un seul seuil.

Figure 3.18 – *Répartitions des régions suivant les seuils utilisés,* $Th_{Cost} = 100$ *pour la ligne du haut et* $Th_{Cost} = 200$ *pour la ligne du bas.*

Méthode de segmentation proposée.

La méthode précédente est fondée sur un critère de distance minimale permettant de fusionner seulement deux régions à chaque niveau de la hiérarchie. Pour résoudre ce problème de complexité, il est possible de relâcher la contrainte sur le nombre de fusions par niveau. Il est ainsi possible d'opérer plusieurs fusions simultanément si la distance entre régions est inférieure à un seuil donné. De même il est possible de s'affranchir de la sur-segmentation sur les contours car, pour une valeur de seuil donné, les régions de petites tailles tentent couramment de fusionner avec les plus larges. Pour cette raison, le concept de distance pondérée a été introduit. Ainsi, contrairement aux approches classiques qui utilisent une distance symétrique, l'approche LAR prend en compte des distances non symétriques entre régions. Une autre amélioration dans le processus de fusion repose sur la définition d'une nouvelle distance basée sur un critère joint moyenne/gradient.

Afin de favoriser l'agglomération des petites régions, la surface des régions est prise en compte via un coefficient de pondération. La distance pondérée est alors donnée par :

$$Cost'\left(R_i^K, R_j^K\right) = Cost\left(R_i^K, R_j^K\right) \log_{10}\left(Surf\left(R_j^K\right)\right) \tag{3.18}$$

avec $Surf\left(R_i^K\right)$ la surface de R_i. Cette dernière propriété implique que le graphe est orienté puisqu'il n'y a pas symétrie dans le cas où les classes considérées n'ont pas la même surface.

Deux métriques sont utilisées pour calculer le critère de choix de fusion ou non de deux régions connexes considérées. La première repose sur une distance simple entre les valeurs moyennes des luminances des régions. La deuxième est fondée sur le calcul de gradient entre les régions. La somme des différences des pixels voisins appartenant à une région différente 2 à 2 permet ici de savoir si une frontière existait entre les régions et s'il est

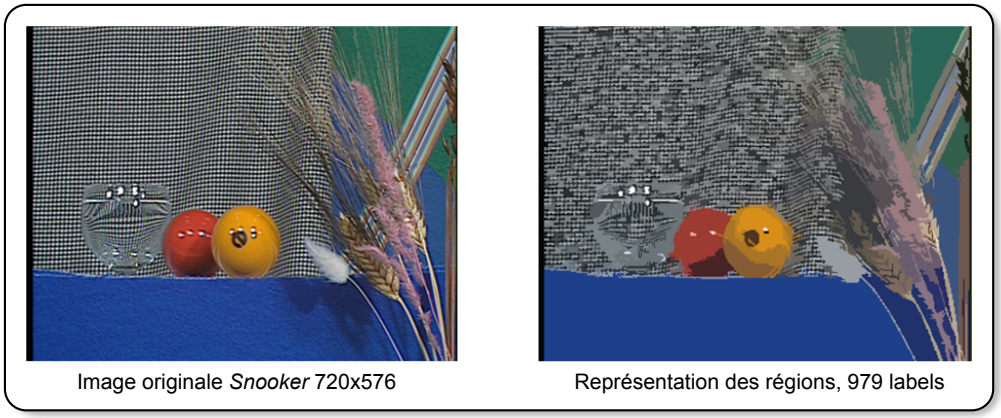


Figure 3.19 – *Répartition des régions sur l'image* Snooker.

légitime de les fusionner. Les deux critères joints ont la même pondération, on a donc le coût de fusion pour le critère luminance :

$$Cost = Cost_M + Cost_{Gr}. \tag{3.19}$$

L'algorithme 3.1 présenté dans [DBBR07] permet de décrire pas à pas la méthode de regroupement des régions à partir de la partition initiale de K_0 régions qui sont encore uniquement des blocs. De manière itérative, l'algorithme procède à la recherche de la région du voisinage la plus ressemblante pour décider ensuite de la fusion ou non suivant un seuil prédéfini Th_{Cost}. Le processus prend fin lorsqu'aucune fusion n'est acceptée par la condition sur ce seuil.

Bien que les outils contenus dans cette fonctionnalité proposée par le codec LAR semblent complets, et que les résultats fournis par la figure 3.18 sont visuellement acceptables, une adaptation à la détection de plages de textures paraissait nécessaire. Ainsi, l'image *Snooker*, illustrée sur la figure 3.19, montre que l'approche initiale ne permet pas de segmenter efficacement la plage texturée occupant le quart de l'image situé en haut à droite. La section suivante décrit les évolutions apportées au schéma afin d'améliorer son adaptation au contexte de notre étude.

3.4.3 Adaptation pour une segmentation au sens de la texture.

Toujours dans l'idée qu'il n'existe pas de segmentation idéale, mais qu'il faut rechercher la meilleure adéquation entre les outils de segmentation et l'application visée, cette section présente les adaptations réalisées.

Prise en compte des statistiques.

La plupart des algorithmes présentés dans le chapitre 1, dont ceux utilisés dans ces travaux de thèse, sont fondés sur une modélisation stochastique Markovienne des textures. La fusion des régions dans la méthode LAR prend déjà en compte le moment d'ordre 1 puisque la valeur moyenne intervient non seulement dans le critère de fusion lié à la luminance mais aussi dans le contrôle par la chrominance. La construction de l'approche en région du LAR n'a donc, au départ, pas la même vocation à détecter des plages de texture partageant des propriétés statistiques commune. L'approche est initialement construite

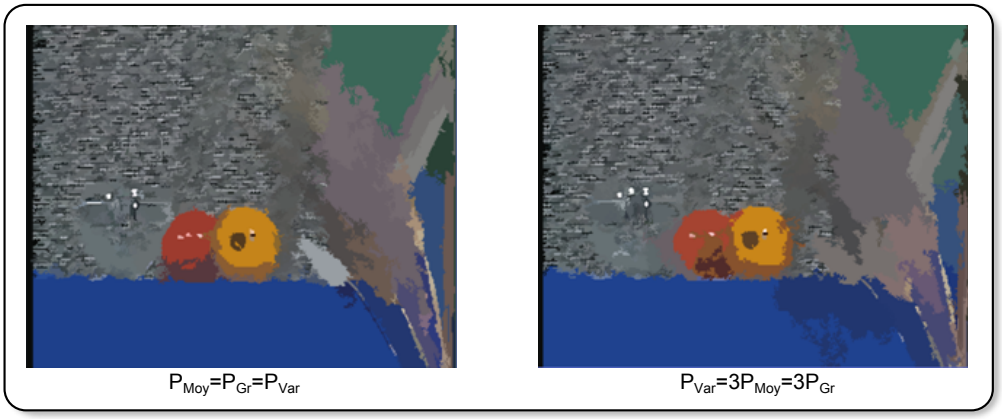


Figure 3.20 – *Régions obtenues sur l'image* Snooker *avec l'ajout de la Variance aux critères joints. La même pondération est appliquée aux critères sur l'image de gauche, une pondération trois fois plus importante pour la variance sur l'image de droite.*

pour rassembler les blocs qui présentent une cohérence en termes de variation de luminosité via le quadtree puis de rassembler les zones avec une luminance moyenne proche. L'idée est donc ici d'ajouter la prise en compte d'un moment d'ordre 2 dans les critères de fusion : la *variance*. Cette dernière, donnée par

$$\begin{aligned} \sigma_X^2 &= \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] \\ &= \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2, \end{aligned} \tag{3.20}$$

traduit la dispersion des variables X du signal par rapport à sa valeur moyenne $\mathbb{E}[X]$. Il paraît donc légitime de fusionner les régions qui possèdent aussi des propriétés du second ordre en commun.

Deux opérations ont été tentées afin de placer ce nouveau critère dans le schéma de fusion des régions.

- Ajout d'une étape de fusion uniquement fondée sur la variance après les étapes successives du schéma actuel.
- Ajout de la variance en tant que critère joint avec $Cost_M$ et $Cost_{Gr}$. On a donc

$$Cost = P_{Moy}.Cost_M + P_{Gr}.Cost_{Gr} + P_{Var}.Cost_{Var} \tag{3.21}$$

avec P_{Moy},P_{Gr}, et P_{Var} les pondérations associées à chaque critère.

Avoir la variance comme critère de fusion nécessite une mise à jour efficace des nouvelles régions formées, au cours des processus de fusion. Ce calcul est facilité par la formule

$$\sigma_{AB}^2 = \frac{N_A(\sigma_A^2 + \mu_A^2) + N_B(\sigma_B^2 + \mu_B^2)}{N_A + N_B} - \mu_{AB}^2, \tag{3.22}$$

où μ est la valeur moyenne des régions et N le nombre de pixels qu'elles contiennent. Cette opération permet d'éviter de calculer à nouveau la variance sur la nouvelle surface.

Expérimentalement, la deuxième méthode a permis d'obtenir les meilleurs résultats sur la plupart des images. Cependant, le but recherché n'est pas atteint avec cette méthode comme le montre l'image de test *Snooker*, qui nous servait d'étalon pour la texture présente en haut à gauche de l'image. La figure 3.20 montre l'évolution des régions en fonction du

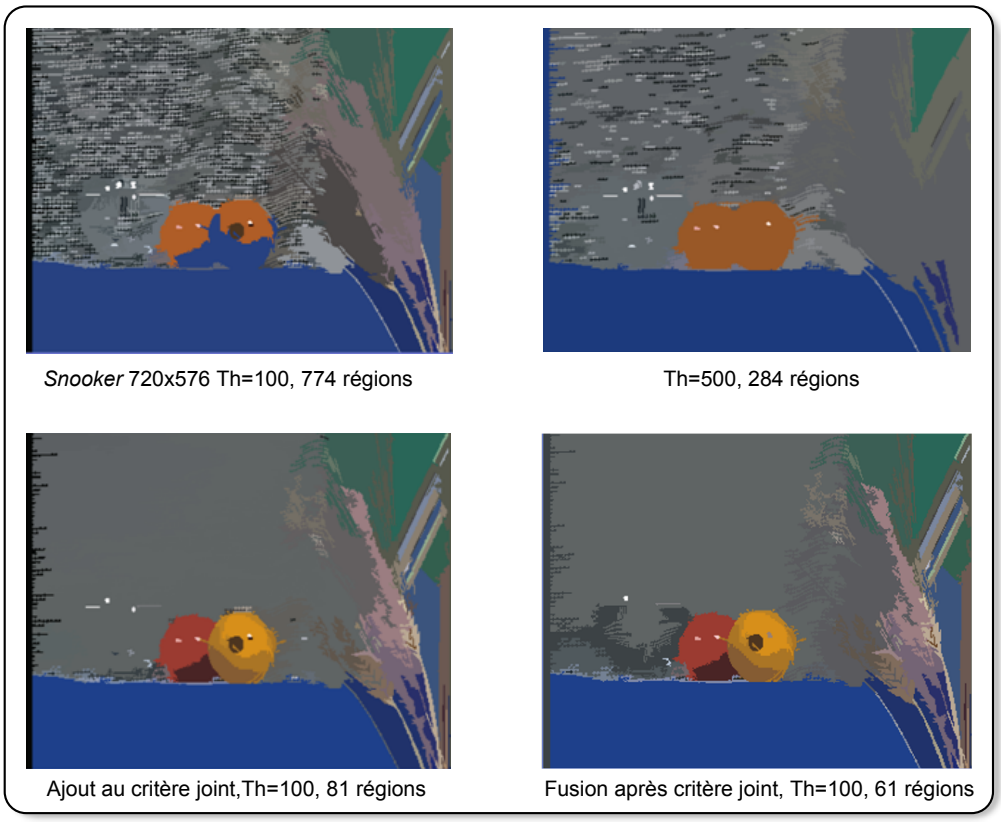


Figure 3.21 – *Répartition des régions sur l'image* Snooker.

poids donné à la variance. On s'aperçoit que cette approche ne permet pas de segmenter correctement la région texturée souhaitée.

Le paragraphe suivant propose alors une deuxième adaptation possible de la segmentation, utilisant les descripteurs DCT, précédemment introduits.

LAR régions et descripteurs DCT.

Dans l'optique d'avoir une segmentation cohérente avec la caractérisation, les descripteurs DCT décrits en section 3.2.2 ont été ajoutés dans le processus de segmentation en régions de la méthode LAR.

Deux mises en place des descripteurs DCT dans le schéma de fusion des régions ont été implémentées. Dans les deux cas, il a fallu déterminer un critère *(Crit)* et un coût *(Cost)* tels qu'ils sont définis dans le LAR basé régions. Pour évaluer ce coût, une distance coefficient à coefficient des descripteurs moyens sur les régions mises en jeu sera calculé, soit mathématiquement :

$$Cost_{Desc}(R_i^K, R_j^K) = \sum_{n=1}^{N-1} \left| D_{R_i^K}(n) - D_{R_j^K}(n) \right|. \tag{3.23}$$

Ce coût fait ensuite l'objet de la même pondération surfacique que pour les coûts orientés moyenne et gradient.

La première implémentation consiste à ajouter cette distance comme troisième critère joint pour les luminances. Le critère devient ainsi :

$$Cost = Cost_M + Cost_{Gr} + Cost_{Desc}. \tag{3.24}$$

La figure 3.21 (image (c)) présente le résultat encourageant obtenu pour la segmentation de la texture souhaitée, en conservant les objets présents. On s'aperçoit que le résultat est nettement meilleur que l'approche région de départ et celle couplée avec le critère de variance. Cependant, le fait d'associer dans une même somme des quantités de nature différente parait compliquer à gérer en termes de pondérations. L'approche du critère joint avec le terme de moyenne et celui de gradient ne fait intervenir que des distances de valeurs de pixels, normalisés par le nombre de pixels mis en jeu. Si le résultat présenté sans pondération sur la figure 3.21 montre que l'approche est prometteuse, il n'en reste pas moins délicat de pondérer ce nouveau critère avec les autres.

C'est pourquoi l'implémentation retenue décorrèle les critères. Une étape de fusion supplémentaire après celles déjà effectuées mais avant la fusion des petites régions est proposée. Ainsi, seul le critère de distances des descripteurs moyens entre en compte pour cette décision de fusion. L'image (d) donnée sur la figure 3.21 montre que pour l'image *Snooker*, cette technique permet d'obtenir les meilleurs résultats. Cette propriété a ensuite été validée visuellement sur un échantillon d'images provenant de séquences notamment utilisées pour les tests des schémas de codage développés dans la suite de ce manuscrit.

Travaux restants.

Même si la segmentation en régions de textures cohérentes, complètement automatique, parait inaccessible, il serait nécessaire de diminuer le nombre de paramètres et seuils à déterminer pour une segmentation acceptable. La recherche empirique des seuils sur les images et séquences de test, présentées dans ce manuscrit, a néanmoins permis de déduire un triplet de seuils de fusion, fonctionnant le mieux en moyenne. Ainsi, le triplet $50, 70, 90$ semble donner des résultats visuellement cohérent sur une large gamme d'images.

3.5 Conclusion.

Les deux thèmes présentés dans ce chapitre, qui traitent de la caractérisation et de la segmentation des textures, ont d'abord constitué une étape nécessaire de ces travaux de recherche. Même si elle n'a pas constitué le cœur des travaux plus axés sur la synthse en elle même, cette étape de segmentation/caractérisation est cruciale pour continuer le développement du reste du schéma de codage, des méthodes pratiques ont été privilégiées au départ. Elles ont cependant fait ensuite l'objet de recherches plus approfondies étant donné l'aspect crucial de la bonne détection des limites et des propriétés des textures. Ne possédant pas d'outils directement adaptés aux fins de détecter et caractériser les régions souhaitées, il a donc fallu partir d'outils existant pour leur apporter la cohérence avec le reste du schéma.

- Pour la caractérisation, les descripteurs de Fourier de la littérature ont ainsi été déclinés en descripteurs DCT qui permettent d'avoir une idée de certains paramètres pour la synthèse de texture.
- Pour la segmentation, l'approche est fondée sur un algorithme *split and merge* à partir d'une décomposition en quatree. La méthode LAR et sa fonctionnalité des régions a constitué un socle solide. L'utilisation des mêmes descripteurs DCT pour

la segmentation permet notamment de consolider la cohérence en fonction des textures. Les régions segmentées le sont en partie parce qu'elles possèdent des propriétés fréquentielles communes.

Il reste néanmoins que ces outils demandent à être optimisés ou partiellement remis en cause. Même si, pour rappel, il n'existe pas d'outil parfait et que ces domaines font et feront encore l'objet de nombreuses recherches, une étude plus approfondie de l'état de l'art corrélé à ce contexte paraît nécessaire. La confrontation expérimentale des méthodes existantes avec celles développées semble donc être un des travaux futurs à mener en priorité.

CHAPITRE 4

Schéma de compression intra image.

À la lumière des techniques présentées dans les chapitres précédents, celui-ci présente la première étape de notre schéma de compression orienté synthèse ou raffinement de texture. La performance de codage recherchée réside dans le fait d'allouer un débit différent pour les zones texturées de celui au reste des images.

Dans un premier temps, un schéma orienté raffinement de texture a été étudié au cours de ces travax de thèse. Ce schéma n'a malheureusement pas été retenu au vu des performances. Dans ce schéma, les patchs utilisés comme échantillons source sont encodés avec une qualité accrue par rapport au reste de l'image. Les détails ainsi préservés dans ces patchs sont ensuite propagés grâce à une étape de raffinement de texture. Cette méthode se trouve éloinée d'un schéma basé synthèse de texture, puisque le raffinement consiste en l'ajout de hautes fréquences. Afin de préserver la linéarité du manuscrit en lien avec la méthode orientée synthèse proposée, ce schéma basé raffinement est décrit en annexe A.

C'est l'inverse qui a été choisi, c'est à dire de sacrifier uniquement les régions que l'on est capable de synthétiser et non encoder des patchs avec une qualité supérieure. Cette cartographie se rapproche plus de celles proposées dans les travaux de compressions orientés texture de la littérature [NNBW09a]. Ce chapitre présente la première étape de construction d'un tel schéma, à savoir une méthode dite *intra* qui utilise des outils fonctionnant image par image. Ils ne conviennent donc qu'aux images fixes ou séquences encodées en mode intra. Après avoir présenté la chaîne globale du schéma, les différents outils adoptés et construits seront détaillés.

4.1 Schéma global retenu.

Pour rappel, le but premier de ces travaux de thèse consiste à produire une solution de compression des textures compatible avec les schémas standards de compression. Ainsi le schéma bloc présenté en figure 4.1 contient un codeur et un décodeur qui peuvent être interchangeables, tant qu'ils permettent d'encoder différemment certaines régions des images ou des séquences d'images.

Dans l'ordre, les images ou séquences d'images sont analysées afin de détecter les zones qui pourront être synthétisées ou raffinées au décodeur. Cette étape contient des outils de segmentation et de caractérisation de texture présentés dans le chapitre 3. Elle

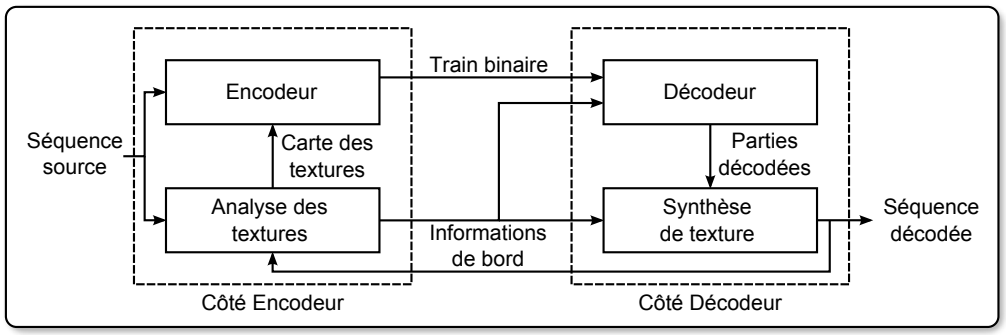


Figure 4.1 – *Schéma bloc du système d'encodage-décodage orienté synthèse de texture.*

fournit une carte des zones texturées synthétisables au codeur. Les zones qui ne sont pas considérées comme synthétisables, qu'elles soient des contours ou les échantillons sources permettant la synthèse au décodeur, sont alors classiquement encodées. De l'information de bord est aussi requise dans certains cas. Il est possible avec le standard H.264 d'encoder différemment les MB d'une même image sans avoir à envoyer d'information supplémentaire pour les détecter au décodeur. Le schéma se voulant compatible avec le plus de systèmes d'encodage/décodage, il peut cependant être nécessaire d'envoyer une version codée de cette cartographie des régions texturées. A l'échelle des MB, cette information codée est de faible coût comparé au reste du débit alloué. Cependant, dans le cas d'une cartographie plus précise au bloc 8×8 ou 4×4 par exemple, il s'agira d'être attentif au coût de cette carte. Il est aussi possible d'envoyer des informations relatives à la synthèse. On peut penser par exemple à la taille des motifs déterminée par la caractérisation de texture présentée en section 3.2.4.

Côté décodeur, les parties encodées classiquement sont d'abord décodées de la même manière. Il reste ensuite à synthétiser les parties manquantes à l'aide de différents outils présentés ci-après.

La section suivante propose de détailler l'analyseur de texture au codeur.

4.2 Analyse de la texture.

Le but de cette étape, précédant l'encodage, est de fournir une étude la plus cohérente possible des textures, cette cohérence étant fondée sur l'ultime étape de ce schéma : la synthèse de texture. Les deux étapes décrites dans le chapitre 3 se succèdent alors comme illustré sur la figure 4.2. Les paragraphes suivants se proposent de détailler ces deux étapes d'analyse.

4.2.1 Segmentation spatiale.

La première étape consiste, comme il est mentionné sur la figure 4.2 de répondre à la question : « où se trouve les régions candidates pour la synthèse de texture ? ». Pour cela, il est nécessaire d'avoir un outils adapté de segmentation au sens de la texture locale. Dans notre étude, la segmentation spatiale utilise principalement l'outil LAR présenté dans le chapitre précédent. Il faut toutefois adapter cette segmentation afin d'avoir une solution compatible avec le codeur utilisé. Les codeurs standards actuels et futurs (HEVC) sont des solutions basées blocs, il convient donc de limiter l'approche à la segmentation sur une

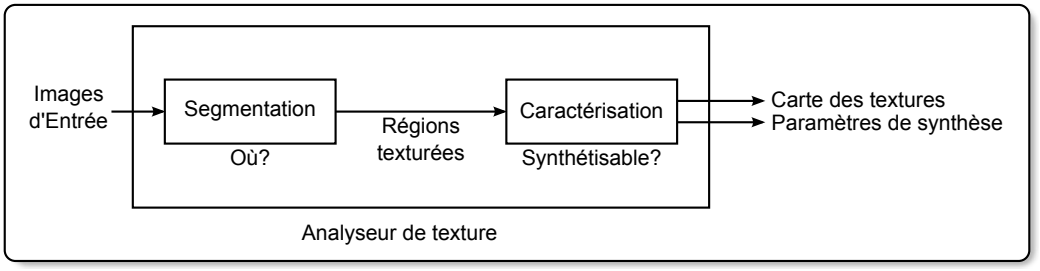


Figure 4.2 – *Schéma bloc du système d'encodage-décodage orienté synthèse de texture.*

grille uniforme de résolution 16×16 pour les MB d'H.264 ou 8×8 pour les blocs JPEG par exemple. Pour ce faire, les étapes suivantes sont réalisées.

1. L'outil de segmentation basé texture, proposé dans le chapitre 3, fournit une carte des étiquettes à la précision pixellique. Il utilise l'approche « split and merge » avec des critères d'homogénéité basés couleur et descripeurs de texture.
2. La grille uniforme est appliquée sur l'image des étiquettes fournies par la première étape. Si un bloc ne contient que des pixels ayant la même étiquette, il garde temporairement cette étiquette, sinon, il est considéré comme bloc de structure et est assigné à l'étiquette 0. Cette étape est illustrée par la figure 4.3 dans laquelle une grille 16×16 a été choisie. Matérialisée en blanc sur l'image (c), elle permet de déterminer les blocs de structures représentés éclaircis sur l'image (d).
3. L'étape suivante consiste en la recherche des régions formées de blocs les plus vastes, à partir de l'image des étiquettes par bloc.
4. Une fois ces régions formées, la mise à jour des étiquettes permet d'obtenir la carte finale, à la résolution dépendant de la taille de bloc choisie.

Après les phases de segmentation et d'étiquetage, l'étape 3 est la plus délicate à adapter. Il faut en effet découper les régions dédiées à être reconstruites au décodeur. Afin d'obtenir des régions aisément synthétisable, il a été décidé de découper des formes rectangulaires comme illustré sur la figure 4.4. Le fait d'encoder différemment certains blocs altère l'efficacité du standard utilisé. Il convient donc de trouver un compromis entre la surface des régions découpées et les hausses de débit occasionnées dans le reste des images. La figure 4.4 présente deux cas de figure.

- Dans le premier, seule la capacité à détecter un rectangle à partir d'une position a été implémentée. Il est possible, comme c'est le cas sur la figure, que les formes des régions fournies par la représentation en régions induisent des rectangles allongés. Cette configuration n'est pas acceptable puisqu'elle ne minimise pas le nombre de MB à encoder alors que les blocs voisins de leur passé causal sera retiré. Leur prédiction s'en trouvant altérée, ces MB seront encodés de manière moins efficace.
- La carte 2 illustre l'option choisie, *i.e.* la détection dans chaque zone du rectangle le plus large afin de maximiser la surface enlevée par rapport à ses MB voisins.

C'est à l'intérieur de ces régions que seront calculés les descripteurs.

4.2.2 Caractérisation de texture.

La description de la caractérisation de texture fournie dans ce paragraphe ne considère que la caractérisation côté codeur, *i.e.* avec la référence à la source. Cette étape qui consiste

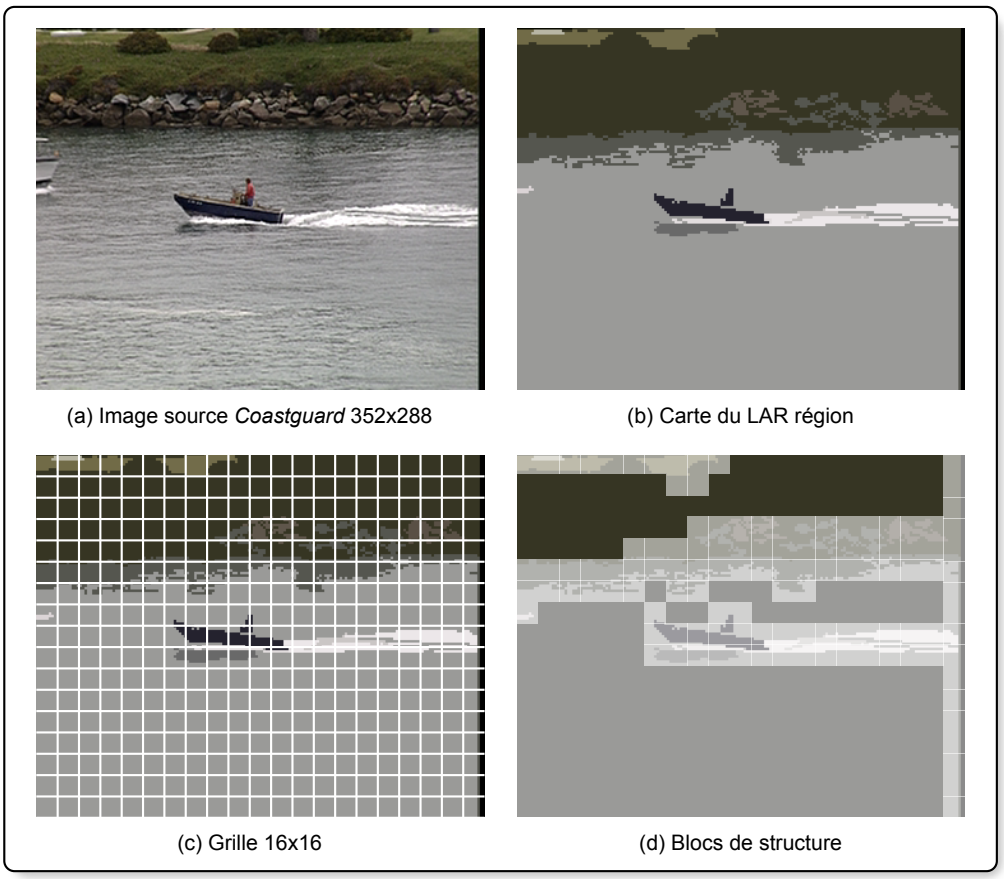


Figure 4.3 – *Adaptation de la segmentation d'une image de la séquence CIF* Coastguard *sur une grille* 16×16.

à déterminer la taille des motifs caractéristiques de texture contenus dans la zone segmentée présente les deux objectifs principaux suivants.

- Fournir le paramètre de taille de motif comme information pour la synthèse. Cette dernière peut être réalisée au codeur pour tester si elle ne produit pas d'artefacts gênants. Il est aussi possible de l'envoyer comme information supplémentaire, encodée dans le train binaire transmis par le codeur.
- Répondre à la question posée sur la figure 4.2, *i.e.* détecter s'il est possible de synthétiser la texture suivant la taille des motifs trouvée. Les échantillons de texture utilisés comme patch pour la synthèse seront décrits plus tard. Cependant, ils sont souvent de taille réduite, il faudra donc que la taille des motifs n'excède pas ces dimensions afin que le patch contienne suffisamment d'informations pour la synthèse.

Les descripteurs DCT présentés dans la section 3.2 sont utilisés à ces fins. La détection des paramètres suit ainsi le cheminement décrit dans la section 3.2.4. Les descripteurs sont calculés à partir de plusieurs tailles de fenêtre sur un large échantillon de positions dans la région candidate à être synthétisée. L'étude de la variation des coefficients des descripteurs basés DCT permet ensuite de définir une taille de fenêtre minimum. Plusieurs cas peuvent apparaître ensuite.

- si cette taille est cohérente avec le reste du schéma, c'est à dire inférieure à la taille du patch choisi pour reconstruire la région, typiquement 16×16, alors la région est

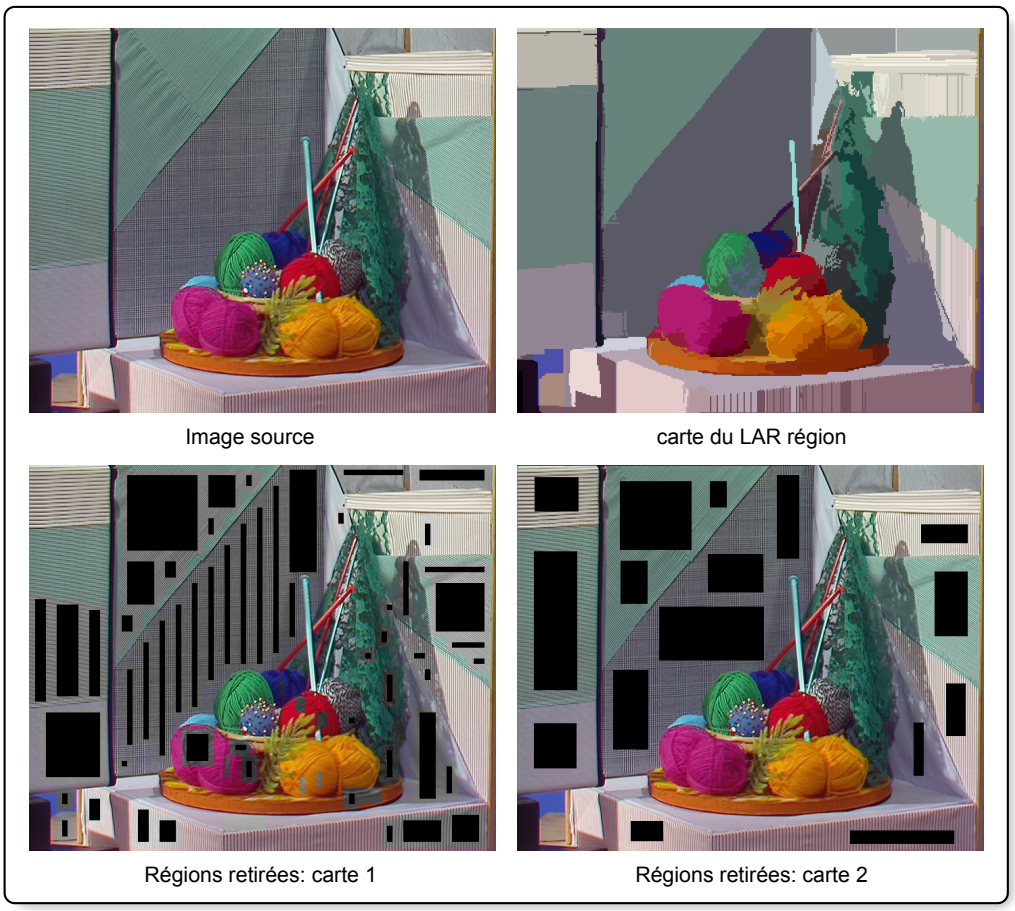


Figure 4.4 – *Segmentation des régions synthétisables.*

synthétisable.

- Si toutes les tailles de bloc mènent à un descripteur décroissant (*cf.* section 3.2), alors nous sommes en présence de textures soit de taille de motif supérieure à 32×32, soit de régions quasi homogènes. Pour le premier cas, la segmentation proposée ne fusionne pas de telles régions. Le second cas est considéré, la synthèse de texture orientée pixel permet d'obtenir pour ces régions de bons résultat avec une grande taille de fenêtre.
- si la taille de bloc minimum est définie mais de taille supérieure au patch, soit 32×32, alors la région n'est plus considérée comme synthétisable, elle est classiquement encodée.

Dans les deux premiers cas, l'algorithme de synthèse doit connaître le paramètre de taille à utiliser au décodeur. Deux solutions s'offrent à nous :

- réaliser l'opération au décodeur sur les zones de la région décodée pour servir de patch,
- transmettre cette information dans le train binaire.

La première solution a d'abord été testée du fait de son avantage de ne nécessiter aucun débit supplémentaire. Cependant, dans la plupart des cas, les tailles de fenêtre déterminées ne correspondent pas au critère choisi pour près de 50% des régions en moyenne sur les séquences *Coastguard* et *Container*. Il a donc été décider de transmettre l'information.

Pour cela, elle est encodée avec les autres tailles de voisinage pour minimiser l'impact sur le débit. L'échantillon des tailles testé étant constitué de $\{4\times4, 8\times8, 16\times16, 32\times32\}$, cette information, avant codage entropique, ne nécessite que 2 bits d'information par région.

La section suivante présente les deux familles d'algorithmes de synthèse choisies.

4.3 Choix des algorithmes de synthèse.

La multitude d'algorithmes de synthèse présentés dans le chapitre 1 ouvre un grand nombre de pistes possibles en vue de construire un schéma de compression. On note toutefois qu'un nombre restreint d'entre eux permettent de synthétiser une large gamme de textures. Les approches de compression de la littérature, présentées en section 2.7, prennent le parti d'utiliser un seul type d'algorithme, censé capable de synthétiser toutes les textures naturelles segmentées dans les vidéos sources. On peut citer l'approche de P. Ndjiki-Nya [NNHW07] dans laquelle la synthèse des textures déformables est assurée par une approche inspirée des travaux de V. Kwatra [KSE+03].

Cette section vise ainsi à développer le cheminement des choix réalisés pour la synthèse contrainte par le contexte spatial et temporel lié à la segmentation de régions dans les séquences naturelles.

4.3.1 Étude des solutions basées pixel.

Les résultats visuels ainsi que la souplesse d'adaptation des algorithmes de synthèse présentés dans la littérature nous ont d'abord incité à s'orienter vers les algorithmes fonctionnant par construction non-paramétrique, *i.e.* les méthodes basées pixel et patch présentées dans les sections respectives 1.5 et 1.6. À cette fin, les approches orientées pixel ont d'abord été implémentées telles que les synthèses de Wei [WL00], Ashikhmin [Ash01], la k-cohérence Tong [TZL+02] ou encore l'utilisation de convergence EM de Kwatra et al. [KAK05] et de Huang et al. [HTW07].

Les résultats visuels ne suffisent malheureusement pas à déterminer l'approche idéale pour la compression. La capacité des algorithmes à s'adapter au contexte spatio temporel des régions à synthétiser est primordiale. L'initialisation aléatoire des données, par exemple, limite la convergence vers une synthèse cohérente avec les pixels déjà décodés. Ainsi, l'approche de M. Ashikhmin nécessite l'initialisation des pixels à synthétiser à partir de valeurs prises dans le patch. La synthèse étant réalisée sur un échantillon aléatoire restreint de pixels candidats, la sélection du meilleur pixel correspondant ne permet souvent pas de compenser la limitation due au nombre réduit de candidats. Cependant, cette limitation peut être atténuée en utilisant conjointement l'algorithme de k-cohérence qui présente néanmoins l'inconvénient de nécessiter une étape supplémentaire de prétraitement pour chaque région segmentée.

Les méthodes les plus évoluées telles que la synthèse EM donnent quant à elles des résultats visuels prometteurs. Les deux algorithmes de synthèse EM [KAK05] et [HTW07] ont alors été implémentées. Cependant les tentatives d'adaptation aux contraintes spatio-temporelles nous ont contraints à commencer par tester des approches plus basiques pour notre schéma. Plusieurs contraintes n'ont en effet pas été levées.

- Tous les pixels des voisinages carrés entrent en jeu dans les calculs de l'étape E, exposés en section 1.7.2. Notre implémentation fournit des résultats visuels satisfaisant, même pour les petits motifs, à partir d'une taille de voisinage de 16×16. Aussi, aucun résultat en image de la littérature ne provient d'une synthèse utilisant

uniquement des voisinages plus petits. Cette propriété contraint donc la taille du patch à contenir un grand nombre de fenêtres 16×16, contrairement aux techniques du type Wei, qui elles se basent sur des voisinages causaux du type 15×8. De plus les voisinages de comparaison peuvent être coupés tant que la forme des voisinages comparés est identique. Dans l'optique de la compression vidéo, la nécessité de préserver des patchs de taille plus grande, transmis au décodeur présente donc une première limite.

- La transition entre les pixels décodés et les pixels synthétisés ne converge pas rapidement, ce qui produit des artefacts gênants aux frontières des régions.
- Enfin, la complexité du système de coordonnée et la détermination d'arrêt de convergence d'un algorithme itératif constituent aussi des problèmes à palier pour la mise en place de tels outils.

4.3.2 Solution basée patch.

Afin de compléter le schéma et de pouvoir mesurer notre approche par rapport aux solutions existantes, l'approche de V. Kwatra [KSE+03] a aussi été étudiée. En effet, les approches les plus abouties présentées dans [ZSWL08] et [NNHW07] utilisent cette méthode de synthèse. Cette technique permet une synthèse plus rapide et plus efficace dans d'autres cas que les approches pixels. Dans certains cas cependant, comme il sera montré dans la section 4.5, la synthèse orientée pixel crée moins d'artefacts et génère un résultat visuellement meilleur. L'algorithme de V. Kwatra a donc été implémenté et adapté au contexte par les méthodes décrites dans les paragraphes suivants, ces méthodes étant compatibles avec les approches patch et pixel utilisées. On verra notamment que l'ordre de génération de la surface diffère. Aussi, par rapports aux résultats présentés dans la littérature, le contexte impose des tailles de patchs à ajouter nettement moindres, étant donné la taille des patchs sources restreinte par la volonté de préserver un maximum de débit. Contrairement à l'approche de [ZSWL07], des patchs de taille supérieure à 8×8 sont considérées, permettant des chevauchements laissant une large place à l'étape de couture.

La section suivante vise à montrer que ces deux types d'algorithmes peuvent être adaptés quasiment de la même manière au contexte de la compression. Ils peuvent alors être utilisés en complémentarité pour synthétiser différentes textures à l'intérieur d'une même séquence ou d'une même image.

4.4 Synthèse de texture adaptée aux régions.

Cette section présente les évolutions apportées aux outils de synthèse afin de les adapter à la compression de régions. Plusieurs outils sont mis au point à cette fin : une carte de confiance permet un ordre de synthèse adéquat, plusieurs tailles de voisinages ainsi que la détermination des patchs sources pour la synthèse. La figure 4.5 permet d'illustrer l'agencement du synthétiseur suivant qu'une étape de caractérisation côté décodeur permet d'obtenir les paramètres nécessaires, ou si ces derniers ont été directement transmis par le codeur.

4.4.1 Ordre de la synthèse.

Les deux types d'algorithmes de synthèse de texture utilisés dans ces travaux de thèse proviennent des approches orientées patch et pixel, décrites dans le chapitre 1. Ces méthodes fonctionnant par construction pixel à pixel, ou patch à patch de la surface à

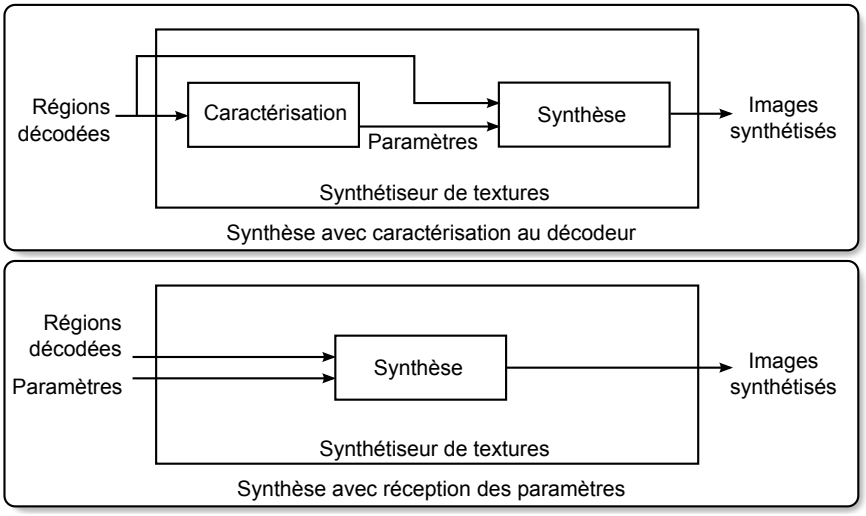


Figure 4.5 – *Schéma bloc de la synthèse réalisée au décodeur. Dans le premier cas, une étape de caractérisation permet de déterminer les paramètres grâce aux zones déjà décodées. Dans le second, ces informations sont reçues comme information supplémentaire.*

synthétiser, il est nécessaire de déterminer dans quel ordre les pixels seront ajoutés. Dans le cas de la synthèse de texture d'une surface rectangulaire à partir d'un patch plus petit, aucune contrainte de bords n'apparait. La question de l'ordre importe peu, même si les travaux de Wei et Levoy [WL00] se démarquent de ceux d'Efros et Leung [EL99] entre autre par l'ordre de la synthèse. Cependant, les pixels voisins des pixels déjà décodés doivent ici être cohérents avec leur entourage. La synthèse dans l'ordre *raster scan* proposée par Wei ne convient donc pas. En effet, comme il est illustré sur la figure 4.6, on remarque que les pixels des bordures, mis à par ceux du haut, ne sont pas ou peu pris en compte dans le processus de synthèse. Un ordre en spirale partant des bordures, comme illustré en figure 4.7, a donc été envisagé afin que les voisinages contiennent au départ les pixels connus. On remarque sur la figure que l'amélioration au niveau des bordures est nette. Cependant de nouveaux artefacts sont créés, notamment des frontières diagonales correspondant aux positions auxquelles la synthèse change de direction.

Il a été finalement décidé d'implémenter une carte de confiance à l'instar de l'ordre proposé dans [CPT04]. Le but ici est d'exploiter à chaque moment de la synthèse les données dans lesquelles on peut accorder le plus de confiance. Au début de la synthèse, les pixels décodés autour de la région à construire relèvent d'une confiance maximum. Ensuite, pendant la synthèse, la confiance accordée à la valeur d'un pixel va dépendre de celle de son voisinage, c'est à dire les pixels qui ont permis de lui affecter cette valeur.

La construction de la carte est illustrée par la figure 4.8, où sont détaillées les étapes d'initialisation et de mise à jour à chaque étape de la synthèse. Le but étant de classer les pixels par ordre de confiance, une liste mise à jour à chaque élément ajouté, pixel ou patch, contient les pixels susceptibles d'être les prochains à être synthétisés. Cette liste contient ainsi les pixels ayant au moins un voisin direct connu, précédemment décodé ou synthétisé.

Comme pour la caractérisation et la segmentation, la manière de calculer la confiance en un pixel se veut cohérente avec l'algorithme de synthèse utilisé. Pour la méthode pixel, le calcul de la confiance est ainsi calculé grâce au voisinage utilisé pour la synthèse alors

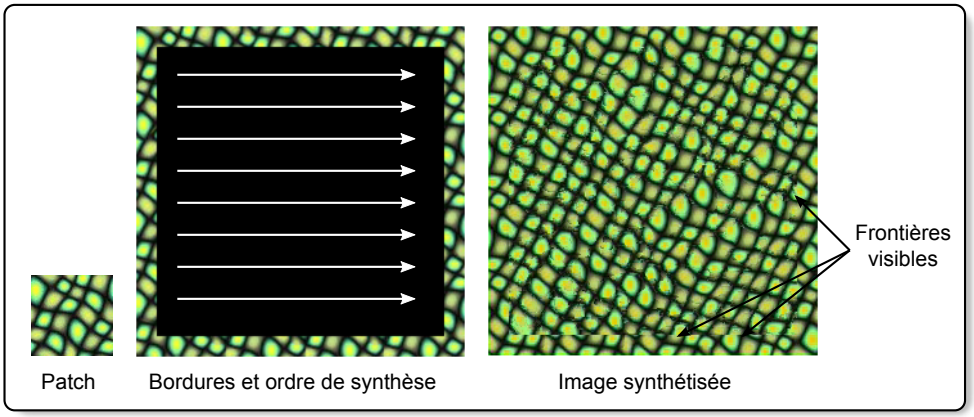


Figure 4.6 – *Synthèse à partir des bords dans l'ordre raster scan.*

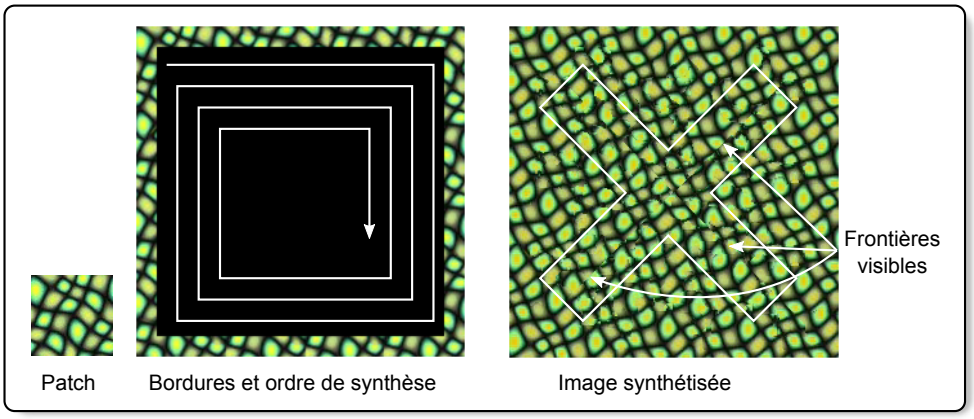


Figure 4.7 – *Artefacts produits lors de la synthèse en spirale.*

que celui-ci sera calculé sur la zone de chevauchement pour la méthode patch. Quelle que soit la méthode, le calcul suit le même raisonnement.

Pour l'initialisation et la mise à jour après synthèse d'un pixel ou patch, illustrées sur la figure 4.8, la confiance C de chaque pixel est issue de la moyenne des confiances sur le voisinage de taille N donnée par

$$C(i,j) = \frac{1}{N^2} \sum_{k=-N/2}^{N/2} \sum_{l=-N/2}^{N/2} C(i+k, j+l). \tag{4.1}$$

où N représente la taille du côté du voisinage. On remarque dans le cadre de la synthèse d'une région rectangulaire que les pixels synthétisés en premier sont ceux des coins qui possèdent le plus de pixels connus. Ensuite, la carte est ensuite mise à jour en assignant la valeur de la confiance calculée du pixel synthétisé $C(i,j)$. On s'aperçoit sur l'exemple donnée par la figure 4.8 que ses pixels voisins ont une confiance accrue.

La carte de confiance pourrait aussi être utilisée à bon escient une version sous résolue sur la surface à synthétiser, même si cette approche ne figure pas dans le schéma final retenu. La version sous résolue peut être ici entendue comme une version sous échantillonnée

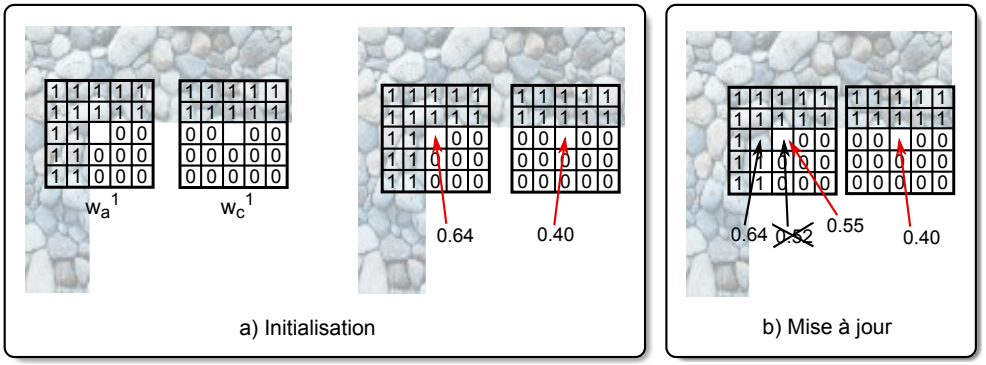


Figure 4.8 – *Ordre de synthèse suivant une carte de confiance.*

spatialement par exemple. Comme il a été présenté dans le chapitre 1, l'utilisation d'un voisinage non causal permet de guider la synthèse dans le cas d'une synthèse pixel. C'est le cas aussi pour la synthèse patch pour laquelle non seulement la zone de chevauchement, mais aussi la zone reposant sur la version sous résolue, seront utilisées pour trouver le meilleur candidat. On parle donc de synthèse guidée ou de raffinement de texture. La carte de confiance permet donc, à l'initialisation, de pondérer la confiance dans les pixels sous résolus différemment des autres pixels décodés. Cette propriété permet notamment d'introduire, lors de la comparaison des voisinages, des pondérations suivant la confiance en chaque pixel du voisinage courant. Cette dernière technique n'apporte cependant pas d'amélioration sensible sur les régions synthétisées, au regard du coût supplémentaire engendré. Elle éloigne aussi de fait l'idée première de l'algorithme de Wei d'utiliser un voisinage de forme et taille fixées au cours de la synthèse.

Enfin, telle qu'elle est implémentée actuellement, cette carte de confiance ne prend en compte qu'une confiance relative aux pixels décodés. On peut cependant penser à pondérer différemment les pixels synthétisés, en fonction de l'énergie calculée lors de la comparaison du voisinage courant avec celui du candidat choisi, ou encore l'énergie de la couture produite dans le cas de la synthèse patch. En effet, plus ces énergies seront faibles plus le pixel ou le patch ajouté seront faibles, plus il sera légitime de leur accorder du crédit pour la suite de la synthèse.

Les figures 4.9 et 4.10 montrent en images les ordres de synthèse fondés sur le maximum de confiance pour la méthode pixel et la méthode patch respectivement. L'aspect pseudo aléatoire du balayage offre peu de zones possibles aux frontières rectilignes, artefacts gênants et rapidement détectés par le SVH. On observe que les blocs présents au début de la synthèse sont utilisés aux moments stratégiques alors qu'ils seraient mal pris en compte suivant un ordre raster ou en spirale. Ces blocs, appelés blocs d'ancrage, seront traités plus en détail dans la section suivante. Cet ordre, outre l'avantage déjà cité, offre la possibilité de traiter simplement différentes formes de régions. Les limitations qui nous ont conduit à considérer des régions rectangulaires dépendent cependant de la question des patchs sources pour la synthèse, aussi discutée dans la section suivante.

4.4.2 Choix de la forme et la taille du patch source.

Dans le cadre de la synthèse de texture, telle qu'elle est présentée dans le chapitre 1, un patch source rectangulaire est utilisé. Ainsi, il est possible de découper un ou plusieurs

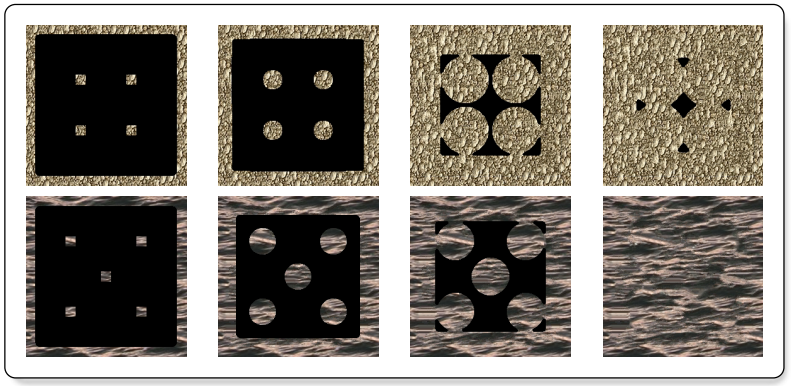


Figure 4.9 – *Illustration de l'ordre de la synthèse de l'algorithme orienté pixel suivant la carte de confiance.*

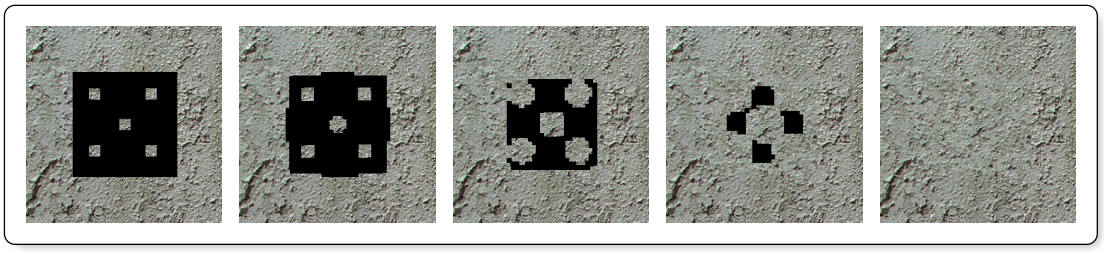


Figure 4.10 – *Illustration de l'ordre de la synthèse de l'algorithme orienté patch suivant la carte de confiance.*

patchs de taille réduite rectangulaire à l'intérieur de la région segmentée, qui sera classiquement encodé comme le reste de l'image, afin d'être utilisé comme échantillon source. Cette opération a été testée et le résultat est présenté sur l'image (a) de la figure 4.11. Les outils utilisés copient les valeurs des pixels ou groupes de pixels choisis du patch sur la surface à synthétiser. Cependant, comme c'est le cas sur l'image (a), les textures naturelles observent souvent de légères variations de luminance et chrominance globale. Même si dans la recherche du meilleur candidat, l'algorithme va tenter de minimiser les erreurs, il ne pourra parfaire la transition avec les bordures en ayant que les valeurs de couleur du patch découpé en haut à gauche de la région.

Plusieurs solutions s'offrent alors à nous pour palier ce problème.

- Utiliser plusieurs patchs afin que les échantillons sources possèdent le maximum d'informations sur la région texturée.
- Utiliser des algorithmes permettant de modéliser les faibles variations d'éclairage, et de recréer une région cohérente. Les techniques présentées dans [XSW10] paraissent prometteuses à cette fin.
- Utiliser comme patch une couronne de pixels tout autour de la région à synthétiser, afin d'être certain d'avoir des pixels candidats proches de toutes les positions de la région. Le terme proche vaut pour l'aspect géométrique mais surtout des valeurs des pixels à synthétiser.

La dernière solution a été retenue pour son efficacité. Le plus souvent, afin de se conformer au standard H.264, la couronne retenue est large de 16 pixels et est ainsi constituée

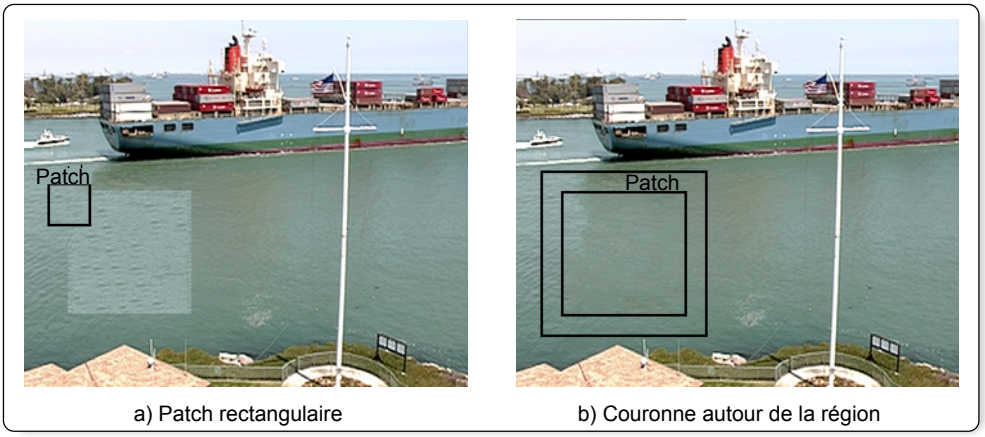


Figure 4.11 – *Forme des patchs a) patch rectangulaire découpé dans le voisinage. b) forme choisie permettant de contenir des pixels cohérents aux 4 coins de la région.*

de macroblocs. L'évolution principale, dans le domaine spatial, sera d'implémenter une solution permettant de retirer des régions ayant une forme quelconque. Une tentative d'implémentation par bloc n'a pas permis à ce jour d'en évaluer le potentiel étant donnée la complexité du patch retenu. Il est en effet nécessaire de définir un ordre de parcours du patch pour rechercher les meilleurs candidats. Nous avons vu en effet dans le chapitre 1 que dans le cas de la synthèse orientée pixel, les bords de patch limitaient les candidats possibles pour la synthèse en fonction de la taille de voisinage choisie. La solution proposée dans [ZSWL07] applique une méthode basée patch sans se soucier de garder un patch spécifique. Cette méthode utilisant des petits blocs de taille 8×8 recherche le bloc 8×8 le plus proche en termes de distance spatiale et de signal, mais aucune localisation précise du meilleur bloc n'est appliquée. Il est donc nécessaire de préserver un « damier » de blocs 8×8 dans les régions texturées pour avoir de bons candidats à chaque position. Dans le cas de la synthèse basée pixel, cette méthode n'est pas efficace puisque les pixels candidats, pour être recopiés, doivent avoir un voisinage autour d'eux contenant des pixels connus. Or conserver comme patch uniquement des blocs 8×8 limiterait fortement le nombre de pixels candidats pour être copiés : 4 candidats par exemple, avec une taille de voisinage de 7×7.

L'ordre de la synthèse, malgré son optimisation, peut produire des artefacts. La figure 4.12 montre que lorsque la région rectangulaire est étirée, une frontière peut apparaître au milieu, zone de jointure en fin de synthèse. C'est principalement le cas lors de la synthèse basée pixel pour laquelle cette frontière correspond à la dernière ligne de pixels synthétisés. Plus le voisinage utilisé est large, plus l'algorithme va tenter de trouver un compromis puisque le voisinage carré autour du pixel courant contiendra des pixels des deux bords. Les tailles de voisinages étant inévitablement restreintes du fait des tailles de patchs engagées, il a été décidé d'encoder certains blocs à l'intérieur de la région, localisées à des positions stratégiques. Dans la suite, ces blocs seront appelés « blocs d'ancrage ».

Une première solution consiste à positionner ces blocs d'ancrage en fonction des descripteurs calculés. Les localisations des descripteurs les plus éloignés du descripteur moyen, en termes de valeurs des coefficients, sont alors candidats pour servir de blocs d'ancrage. Il s'avère cependant que les artefacts sont dus à la géométrie de la synthèse et non aux positions pour lesquelles le signal diffère du reste. Cette sikution ineficace a donc été aban-

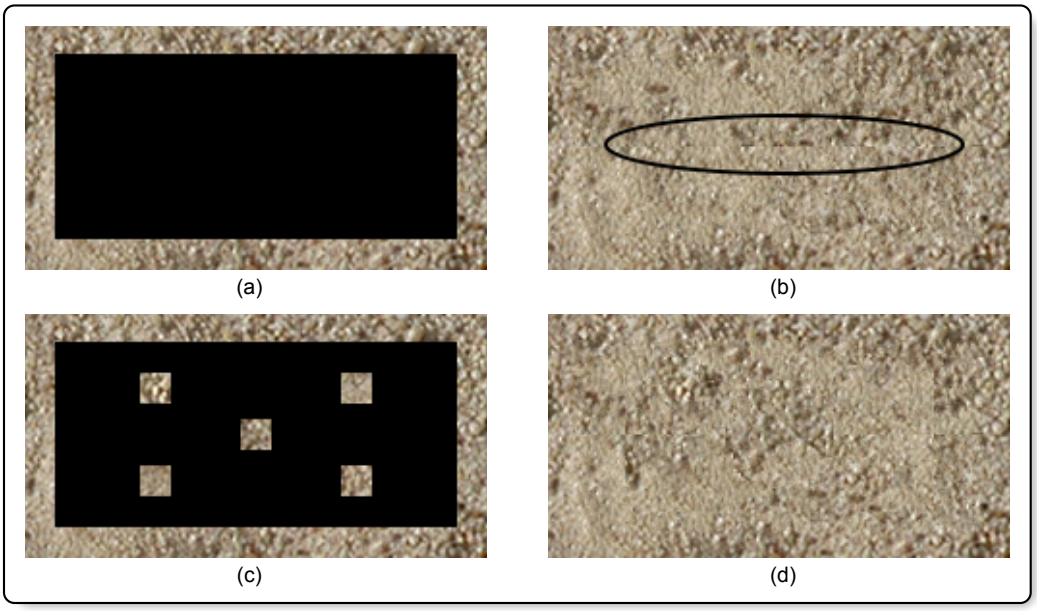


Figure 4.12 – *Préservation de quelques blocs d'ancrage. (a) et (b) : synthèse sans bloc d'ancrage. (c) et (d) : synthèse avec blocs.*

donnée.

Un système de coordonnées géométriques, en fonction de la taille de la région a donc été implanté. Ces positions stratégiques permettent de réduire au maximum les jointures rectilignes, sources de frontières visibles. La figure 4.12 présente l'amélioration constatée pour la synthèse orientée patch, la frontière centrale a disparu sur l'image (d). Ainsi, lorsque la taille de la fenêtre en horizontal ou vertical excède un nombre choisi n MB, le bloc central est encodé. La région est ensuite découpée en 2 rectangles si une seule des composante excède cette taille, en 4 rectangles si la région est assez grande dans les deux dimensions. Les blocs centraux de ces nouvelles régions sont aussi conservés dans le cas où leur dimension est encore suffisante.

La section suivante propose, côté codeur de choisir pour chaque région à synthétiser, quel algorithme choisir.

4.4.3 Décision au codeur de l'algorithme de synthèse.

Une méthode a posteriori est proposée côté codeur. Elle permettra de détecter les artefacts et de valider si la synthèse est acceptable. La synthèse basée patch, moins complexe, est d'abord réalisée avant d'être évaluée par la méthode des gradients décrite précédemment.

Nous avons vu que l'approche patch, telle qu'elle est implémentée, produit des artefacts particuliers quand elle est défaillante. Cependant, ces artefacts ne sont pas localisés qu'aux bordures de la région à synthétiser. En effet, ces défaillances étant dues aux coutures visibles entre les patchs ajoutés, elles se transcrivent à l'œil par des frontières nettes en ces endroits dans toute la région, comme pour les synthèses illustrées sur les images (c) et (d) de la figure 4.1. On note notamment que lorsque la correspondance entre le patch à ajouter et la surface existante est mauvaise, les coutures rectilignes sont privilégiées.

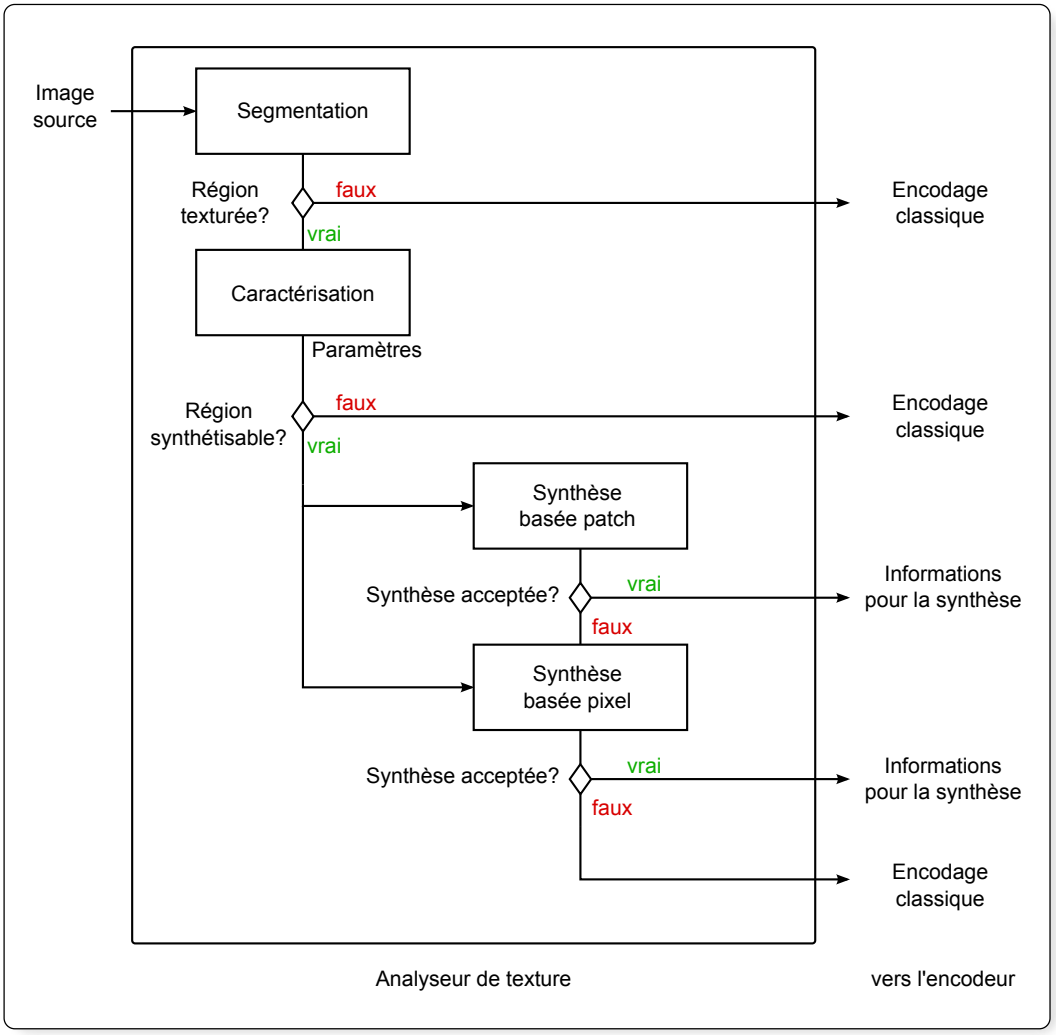
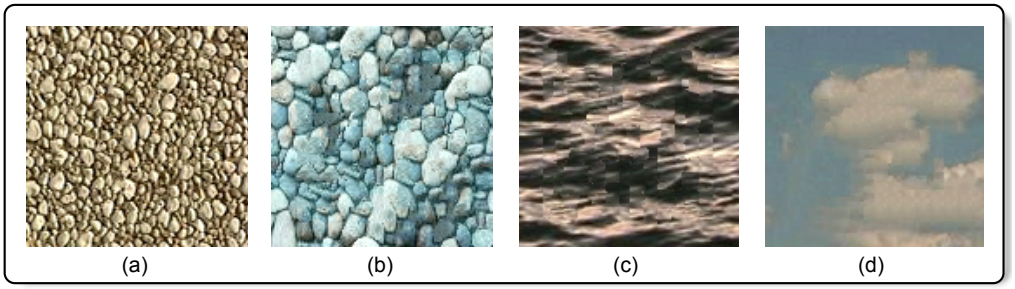


Figure 4.13 – *Schéma résumant les opérations réalisée par l'analyseur de texture côté codeur.*

Afin d'évaluer un maximum d'artefacts potentiels de la synthèse basée patch, nous proposons une triple note orientée patch. Une première note Gl correspond aux différences entre les pixels directement voisins et appartenant avant couture à des patchs différents. Cela permet de détecter si des frontières gênantes ont été créées entre la région d'origine et la région synthétisée.

Les deux autres notes permettent de mettre l'accent sur les artefacts les plus gênants pour le Système Visuel Humain, à savoir les frontières rectilignes horizontales et verticales qui s'apparentent aux effets de bloc. Un gradient de type Sobel est calculé pour chaque pixel de la région synthétisée et la région originale. Les noyaux horizontaux et verticaux sont respectivement donnés par :

$$\mathbf{G_h} = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{I}, \tag{4.2}$$



Texture	ΔGh	ΔGv	ΔGl
a	-0.3	-0.8	0.93
b	-3.5	-4.1	17.6
c	-12.3	11.1	28.7
d	-4.0	15.3	60.7

Table 4.1 – *Gradients obtenus pour la synthèse orientée patch sur des échantillons caractéristiques de texture a, b, c et d illustrés.*

$$\mathbf{G_v} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{I}, \tag{4.3}$$

où I correspond aux valeurs des pixels de l'image considérée.

Après des expérimentations sur un échantillon d'une trentaine de patchs représentatifs des textures naturelles, les seuils $\{5; 5; 20\}$ ont été déterminés pour ΔGh, ΔGv et ΔGl respectivement. Ainsi, dès que l'un de ces seuil est franchi, la synthèse orientée patch est considérée comme défaillante et la synthèse orientée pixel est appliquée.

La décision de l'algorithme choisi étant réalisée côté codeur, puisque l'image source est requise, il faut envoyer l'information binaire au décodeur lui permettant d'appliquer le bon algorithme. Cette information représente 1 bit d'information pour chaque région constituée la plupart du temps de plusieurs MB, cette information encodée dans le train binaire est donc très petite devant le reste du contenu transmis.

4.5 Résultats.

4.5.1 Mise en place des tests.

Le schéma a été testé en combinaison avec le codeur JM [jvt07]. Plusieurs séquences ont servi pour les tests. Les figures et tableaux présentés ne listent qu'un échantillon représentatif, les sources utilisées sont représentées en images dans l'annexe B.

Afin d'évaluer correctement les gains en débits atteints avec un schéma conjoint au standard H.264, une manipulation au codeur est nécessaire. Le mode *skip* correspond dans H.264 au « non codage » du résidu après la prédiction inter d'un MB. Ainsi, le décodeur construit le MB uniquement à partir des prédictions des MB voisins. Ce mode correspond parfaitement au traitement désiré pour les MB qui seront reconstruits par le synthétiseur de texture. Cependant, ce mode n'est disponible que dans le domaine inter image, soit pour les images de type B et P, alors que l'encodage d'une image unique est nécessairement intra. Pour pallier ce problème, nous avons décidé de créer une séquence de deux images.

- La première image est noire, ou du moins unie de la valeur choisie pour les blocs à synthétiser.

QP	10	15	20	25	30	35	40
Sequences	Bit-rate saving (%)						
Coastguard CIF	21,8	22,4	20,7	20,3	19,7	19,5	18,9
Container CIF	21,7	19,3	14,4	10,6	7,1	4,5	2,7
Wool SD	14,4	13,9	11,7	10,3	8,8	7,5	7,1

Table 4.2 – *Gains en débits suivant le QP utilisé.*

- La deuxième image correspond à l'image à encoder.

La séquence est ensuite encodée avec une structure IP. Ainsi, la première image codée en intra nécessite un minimum de débit étant donné qu'elle est unie (les informations relatives au mode de codage DC pour tous les MB). Le codeur va ensuite naturellement choisir les modes intra pour touts les blocs non synthétisés et au mode *skip* pour les blocs noirs, puisqu'ils correspondent exactement au MB colocalisé dans l'image I.

La figure 4.14 ainsi que le tableau 4.2 contiennent des résultats correspondant à la synthèse de régions non encodées, c'est à dire qu'aucune version sous-échantillonnée n'est transmise afin de guider la synthèse. Le tableau 4.2 référence les gains de débit réalisés pour les séquences *Container, Coastguard* et *Wool*. La gamme de QP {10,15,20,25,30,35,40} permet de mesurer l'impact des régions retirées à différents débits. On remarque sur ce tableau que les gains sont logiquement moindres pour les bas débits étant donné que l'approche s'appuie sur le forçage du non codage de résidu, ce dernier étant moindre à bas débit. Il chute notamment dramatiquement aux très bas débits puisque les *skips* forcés se substituent aux *skips* naturels, aucune modification du codage de ces MB n'est alors opérée. Les gains en débit des images utilisées pour les tests subjectifs seront développés dans le paragraphe dédié à ces tests.

Le paragraphe suivant présente une campagne de tests subjectifs menée afin de déterminer l'impact de la synthèse sur la qualité perçue par un panel d'observateurs.

4.5.2 Tests subjectifs.

Il a été discuté en section 2.5 de la non-adaptation des méthodes de mesure objectives au problème de l'évaluation de la qualité des synthèses de texture. En effet, toutes les méthodes du type PSNR, qui comparent les pixels aux mêmes positions entre une image dégradée et une image source, donneront forcément des résultats médiocres pour les images synthétisées. Les pixels reconstruits le sont en fonction de leur voisinage, mais aucunement pour approcher le signal source à une position donnée. Ainsi, même les métriques structurelles, et autres adaptations au SVH ne permettent pas d'évaluer une synthèse.

Mises à part les techniques de détection d'artefacts telles que celle proposée en section 4.4.3 et celle proposée dans [NNKW04, NNHW07], aucune méthode automatique ne permet donc d'évaluer la qualité de la même manière que le PSNR permet d'évaluer une image décodée. C'est pourquoi la plupart des documents de la littérature tentent de fournir les gains en débit réalisé « à qualité visuelle similaire ». Une campagne de tests subjectifs a donc été mise en place afin de mesurer s'il était possible de faire cette hypothèse d'une part, dans le cas de notre approche *intra* image.

Protocole expérimental.

Bien que les expérimentations présentées dans ce paragraphe ne suivent pas exactement les recommandations ITU-R-BT. 500-11 [ITU02], toutes les conditions permettent

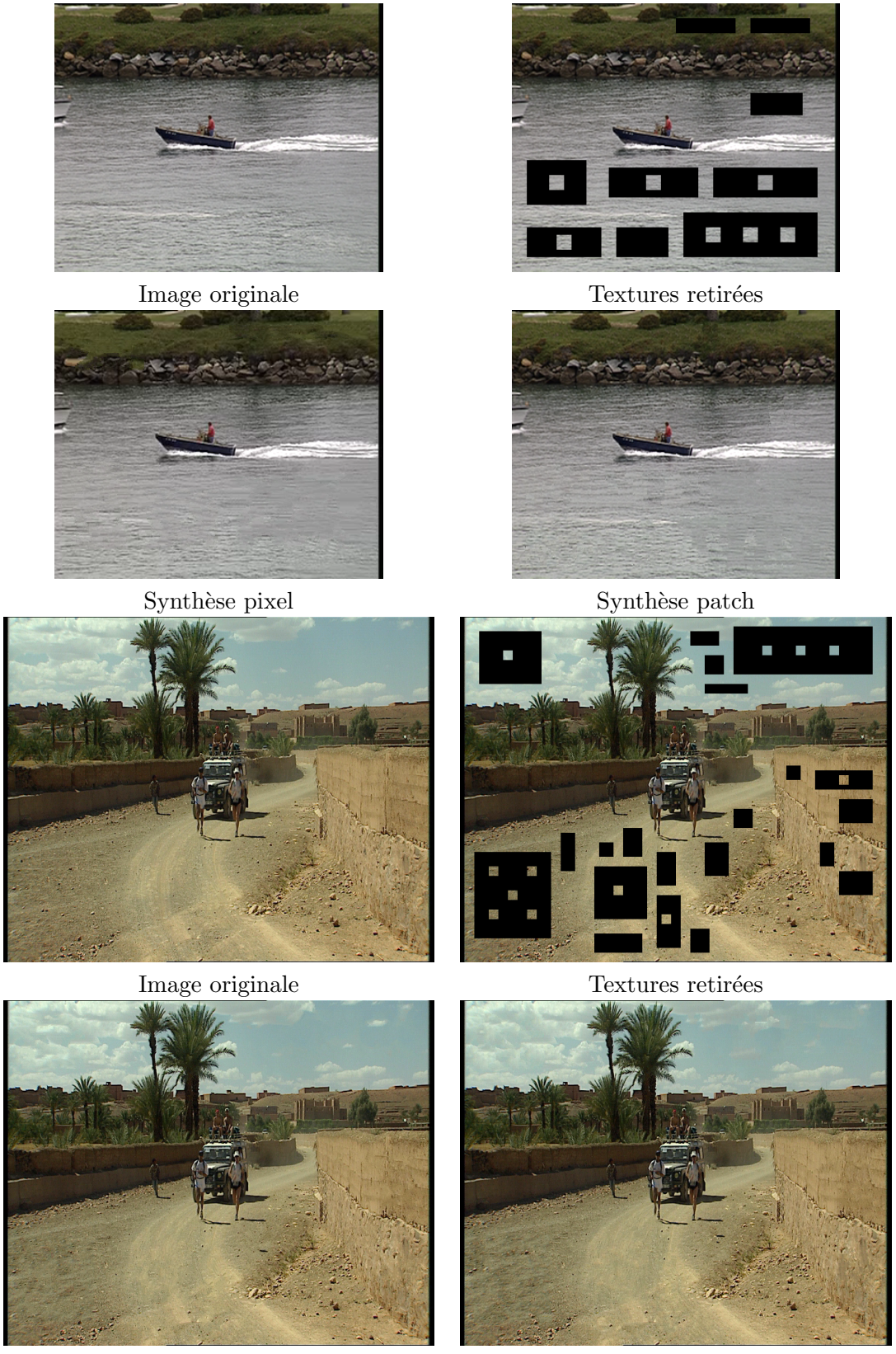


Figure 4.14 – *Résultats en images des techniques basées pixel et patch pour les séquences* Coast-guard *CIF et* Raid Maroc *SD.*

Figure 4.15 – *Images présentées pour l'évaluation subjective du schéma orienté synthèse par rapport aux images décodées par H.264 en mode Intra.*

d'approcher les conditions idéales d'évaluation. Les tests ont été réalisés dans une salle dédiée du laboratoire IETR sur un moniteur CRT. Les observateurs sont placés à une distance environ égale à 4 fois la hauteur de l'image. La luminance du fond de la pièce est réglée de manière à approcher les conditions spécifiées par l'ITU. Les images présentées sur la figure 4.15 ont été évaluées par un échantillon de 16 personnes. Pour chacune des images sont présentées les versions reconstruites par un décodeur H.264 sans synthèse de texture pour $QP = \{10, 25, 40\}$. Pour chaque niveau de quantification, 4 images sont ainsi à évaluer :

- l'image entièrement décodée,
- l'image dont les régions texturées sont synthétisées par la méthode patch,
- l'image dont les régions texturées sont synthétisées par la méthode pixel,
- l'image dont les régions texturées sont synthétisées avec la méthode patch ou avec la méthode pixel.

Pour les trois dernières images, les régions enlevées par l'étape de segmentation sont présentées sur la figure 4.16. Chaque observateur a donc 4×12 images issues de la même image originale à évaluer, soit 48 images en tout. Pour chaque groupe de 12 images, le sujet choisit d'observer, noter, revenir sur les images, et décide du temps d'observation

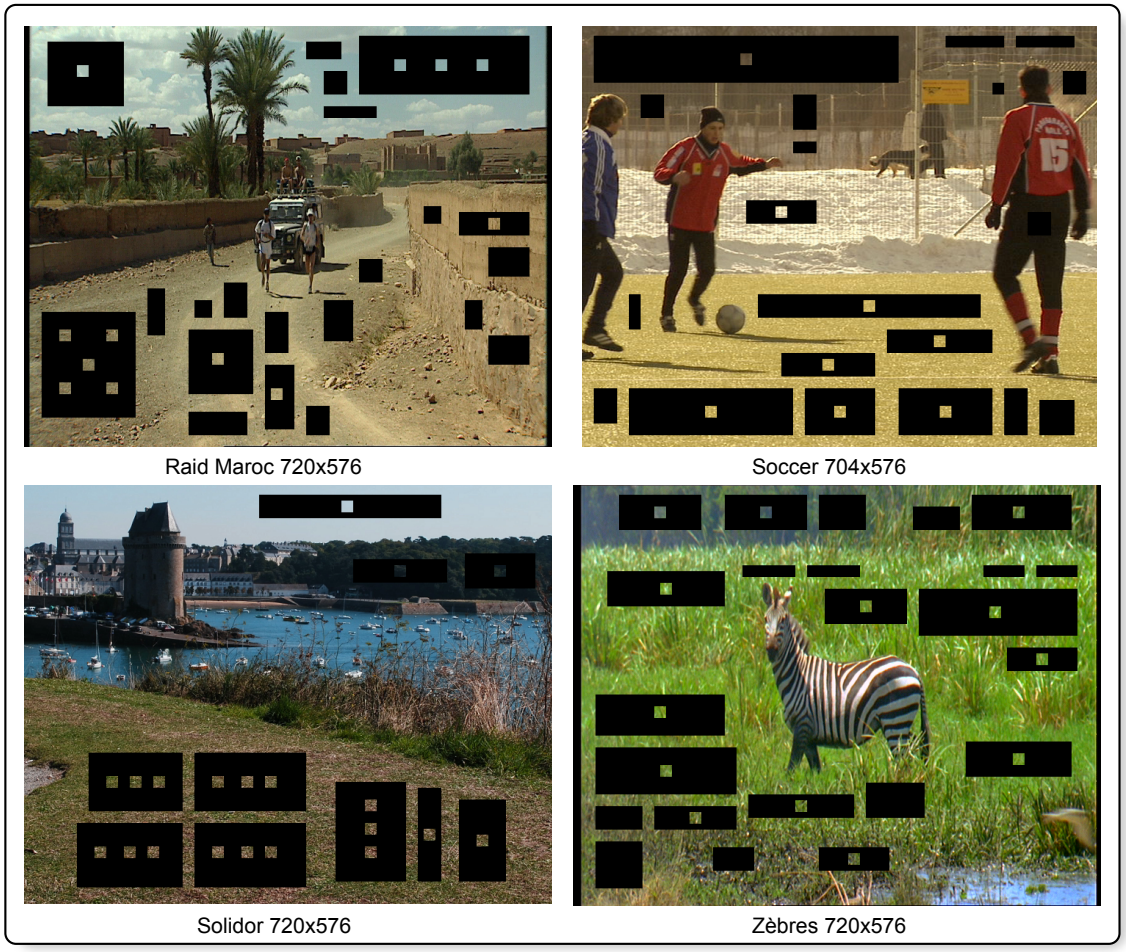


Figure 4.16 – *Régions synthétisées des images évaluées lors de tests subjectifs.*

nécessaire pour l'évaluation. La durée de l'expérience dépend donc directement du temps mis par les observateurs à évaluer les 4 groupes d'images, typiquement 10 à 15 minutes au total.

Les images sont évaluées sur une échelle de 1 à 10, 10 correspondant à la qualité maximale. L'image source n'est pas présentée : l'image de meilleure qualité pouvant servir de référence à l'observateur correspond ainsi à celles décodées avec $QP = 10$. Parmi le panel d'observateur, un seul a été rejeté par l'algorithme fondé sur la méthode du χ^2 le coefficient de corrélation de Spearman. Le paragraphe suivant détaille et analyse donc les résultats obtenus sur l'échantillon des 15 sujets retenus.

Résultats

La figure 4.17 présente les résultats obtenus par QP. On observe premièrement qu'il n'est pas rigoureusement légitime de présenter les gains obtenus comme *gratuits* en termes de qualité puisque les observateurs ont noté une dégradation sensible entre les images décodées classiquement par AVC et les images issues de notre schéma. Il est à noter aussi que la solution basée pixel est préférée dans une large majorité des cas. En fonction des débits observés, la synthèse basée patch fonctionne mieux pour les bas débits, alors que

QP	10	15	20	25	30	35	40
Séquences	gain en débit (%)						
Zèbres	24.2	23.9	23.1	20.1	20.8	16.7	16.7
Solidor	22.0	22.6	23.2	24.1	25.0	26.1	22.9
Raid Maroc	16.1	15.7	15.2	14.2	12.7	2.0	1.7
Soccer	19.8	20.1	20.2	20.4	20.3	18.2	10.5

Table 4.3 – *Gains en débit sur les séquences utilisées pour les tests subjectifs.*

l'approche orientée pixel est plus appréciée pour les hauts débits.

Comme il est décrit dans le protocole, l'observateur a accès à tout moment à une version décodée de très bonne qualité (QP=10). Ce protocole permet une évaluation honnête du schéma puisqu'en comparant une version décodée avec $QP = 25$ et sa version synthétisée, l'observateur peut apprécier les différences et artéfacts introduits par les différents outils utilisés. C'est cependant une méthode d'évaluation qui défavorise la synthèse puisque les différences avec la source apparaissent et focalisent l'attention. Il nous fallait suivre ce protocole afin d'évaluer légitimement l'approche. Il paraît néanmoins important de pratiquer la méthode d'évaluation subjective suivante : au lieu de permettre à l'observateur de naviguer parmi les images décodées pour plusieurs niveaux de qualité, il ne pourra qu'observer la version décodée à un QP donné et sa version synthétisée. L'évaluation d'un nouveau niveau de qualité se fera après l'évaluation d'autres images. L'observateur n'aura ainsi plus accès à une version proche de la source sur laquelle s'appuyer.

4.6 Conclusion.

Ce chapitre a permis de détailler les approches *intra* images développées au cours de ces travaux de thèse. Les limites soulignées du premier type de schéma nous ont conduits à orienter les travaux vers une solution plus proche de celles de la littérature présentées en section 2.7. La multitude d'algorithmes étudiés et implémentés - les approches de Wei et Levoy [WL00], Ashikhmin [Ash01], Tong et al. [TZL+02], Kwatra et al. [KSE+03, KAK05] et Huang et al. [HTW07]- a permis d'extraire les algorithmes les plus aisément adaptables à notre contexte. Cependant, les approches EM, qui n'ont pas été retenues pour ce schéma, restent disponibles pour un schéma futur du fait des performances visuelles de synthèse, et ce malgré leur difficulté d'adaptation.

Le schéma développé utilise deux approches : une orientée pixel dérivée des travaux de L.Y. Wei et une orientée patch inspirée de l'approche de V. Kwatra. Ces deux approches ont vocation à être utilisées en complémentarité, suivant les types de texture de prédilection de chacune. Pour la décision du synthétiseur orienté pixel ou patch, une méthode a posteriori est proposée. La méthode patch étant plus rapide à exécuter, elle est appliquée avec les paramètres directement fournis par les descripteurs issus de l'analyse de la texture. Si l'évaluation par une méthode basée gradient détecte trop d'artefacts visibles, la méthode pixel est alors préférée. Dans la même logique, avec un coût supérieur, il est possible de remettre en cause la segmentation en évaluant la synthèse basée pixel. Si celle-ci s'avère défaillante, la région en cours sera encodée classiquement.

L'évaluation de la qualité de la synthèse constitue le principal obstacle à la valorisation d'une telle approche. En effet, les critères objectifs usuels tels que le PSNR largement utilisé dans le domaine de la compression ou même les métriques de similarité comme la

SSIM ne permettent pas d'évaluer l'impact de la synthèse dans l'image. Ces méthodes s'attachent à déterminer les écarts par rapport à une source d'une manière ou d'une autre. C'est aussi le cas dans la campagne de tests subjectifs menée, où, dans le but de légitimer l'évaluation, une image de référence est proposée à l'observateur. De futurs tests subjectifs, sans référence à une image source, permettront de déterminer si la synthèse n'est pas gênante pour l'observateur et donc si les gains avancés dans ce chapitre sont réellement fondés.

Cette approche limitée au domaine spatial est étendue au domaine temporel suivant le schéma décrit dans le chapitre suivant.

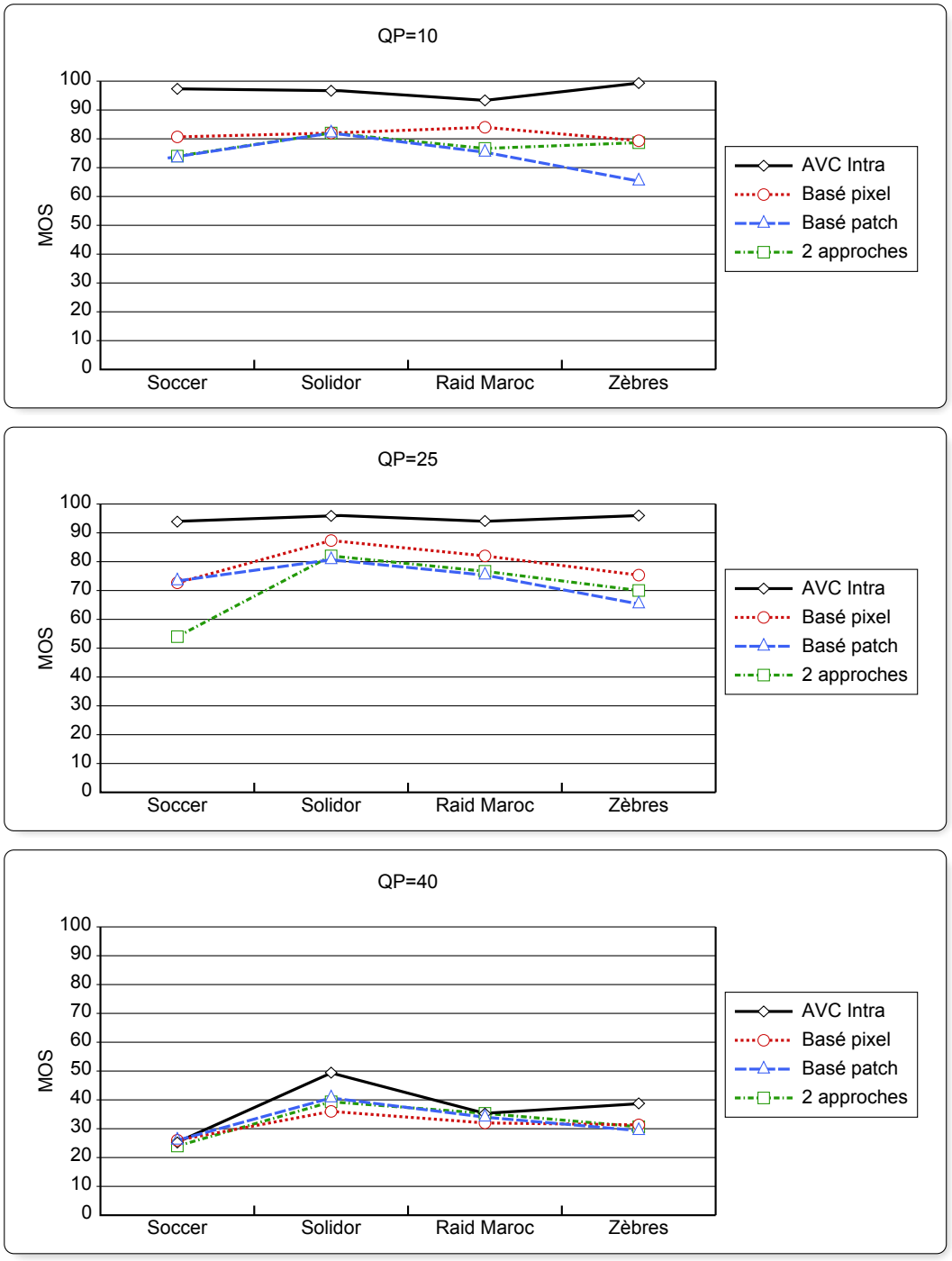


Figure 4.17 – *Résultats obtenus lors de tests subjectifs.*

CHAPITRE 5

Compression et synthèse d'images animées.

Ce chapitre vise à étendre le schéma présenté dans le chapitre précédent au domaine temporel. La principale problématique à ajouter aux contraintes déjà énoncées pour la synthèse dans un schéma de compression intra image, se situe au niveau de la cohérence entre les régions synthétisées d'une image à l'autre. L'estimation de mouvement, dont une forme a été présentée dans le chapitre 2, constitue alors le point de départ pour résoudre les tâches suivantes :

- assurer la cohérence temporelle de la synthèse de texture sur les images successives d'une séquence,
- permettre d'interpoler les pixels d'images intermédiaires à partir d'images de référence et du modèle de mouvement estimé,
- segmenter la séquence ou une sous-partie de la séquence en régions cohérentes au sens du mouvement.

La figure 5.1 permet d'illustrer le nouveau problème soulevé par le traitement de séquences vidéo. Le but du schéma sera ainsi de construire une synthèse vis-à-vis du contexte spatial de l'image courante, mais aussi des images de son environnement temporel. Les images de référence permettront, comme dans les standards décrits dans le chapitre 2, que les modes de codage et de synthèse ne divergent pas temporellement par rapport aux mouvements de la scène observée. On s'aperçoit alors que le problème peut être assimilé à la reconstruction d'un parallélépipède qui aurait été retiré de la séquence.

Afin de conserver une cohérence entre les images synthétisées et le reste de la séquence, des images non synthétisées permettront d'éviter à la synthèse de diverger. Ces images seront appelées **images de référence** dans la suite du chapitre. Ces images n'étant pas forcément corrélées aux images clés du codeur H.264 joint, les séquences seront divisées en GOP pour le standard et GOPM (groupes d'images au sens du mouvement) pour le schéma de compensation/segmentation de mouvement développé dans ce chapitre. Aussi, les images dites de référence délimitent les GOPM. Ces derniers seront composés, d'un point de vue synthèse des textures, des images de référence non synthétisées de part et d'autre, alors que les images intermédiaires, intercalées, pourront faire l'objet de synthèse. Par exemple, si une structure IBPBPBPBI... est choisie pour le codeur, un GOPM pourra être IBPBP. Dans ce cas, l'image I et la deuxième image P ne seront pas synthétisées et serviront de référence pour l'estimation de mouvement par région.

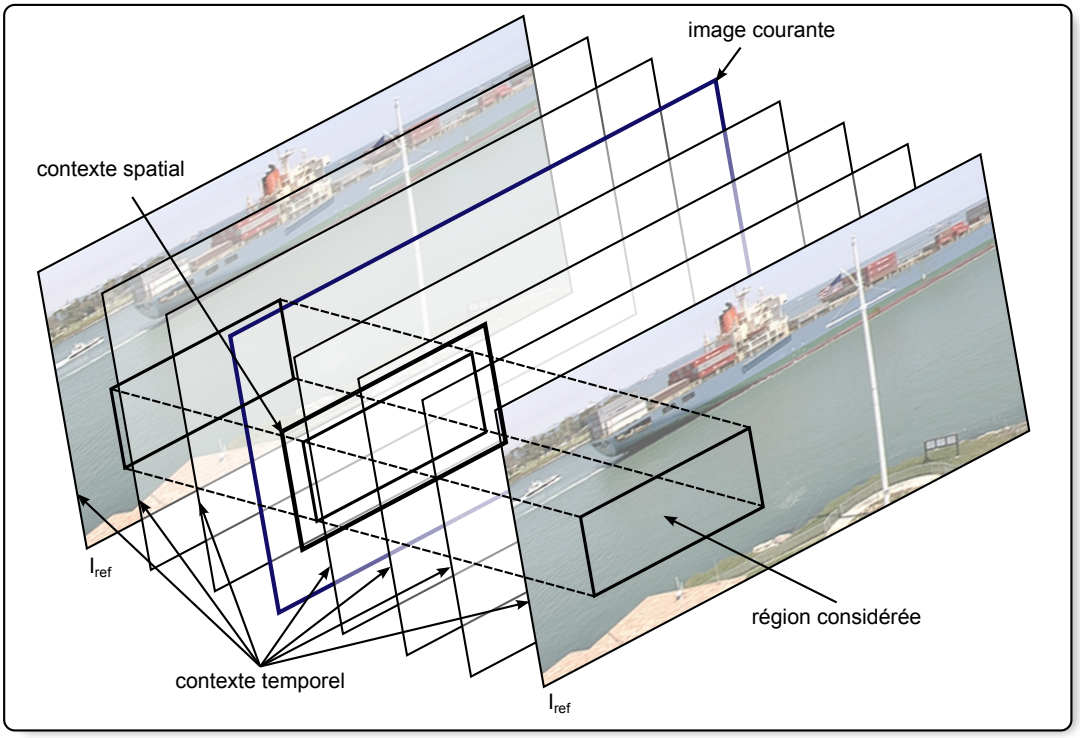


Figure 5.1 – *Contexte spatio-temporel d'une région texturée à synthétiser*

Dans ce chapitre, après avoir décrit la méthode d'estimation de mouvement par régions adoptée, les méthodes de segmentation temporelle et l'adaptation des outils de synthèse seront détaillées. Un dernier type de reconstruction des régions texturées sera présenté pour celles dont le mouvement est correctement modélisé. La synthèse temporelle, fondée sur les algorithmes mis en place pour le schéma intra, permettra de reconstruire les régions texturées dont les mouvements locaux ne permettent pas de définir un modèle global acceptable. Des résultats fondés sur des tests subjectifs permettront enfin d'évaluer l'apporche proposée.

5.1 L'estimation de mouvement.

Cette section vise à présenter les outils d'estimation locale de mouvement et des modèles plus évolués permettant de décrire le mouvement de régions entières grâce à un nombre restreint de paramètres.

L'estimation de mouvement consiste pour cela à modéliser sur un plan 2D le mouvement d'objets 3D ainsi que les mouvements de la caméra. Cette projection du mouvement sur le plan correspond au *flot optique*. Ce dernier est abordé dans les traitements en faisant l'hypothèse de la conservation de la luminance qui vérifie pour deux images consécutives d'une séquence $I(x, y, t)$ et $I(x - v_x, y - v_y, t - 1)$. Ainsi, on émet l'hypothèse

$$I(x, y, t) = I(x - v_x, y - v_y, t - 1), \tag{5.1}$$

où $\vec{v} = (v_x, v_y)^T$ représente le vecteur vitesse à t du pixel aux coordonnées (x, y). Plusieurs phénomènes peuvent mettre cette hypothèse en défaut, comme les occultations ou les changements d'illumination de la scène dus à de nouvelles sources de lumière comme les

flashs d'appareil photo. Des modèles plus fins permettent de tenir compte de ces variations, leur complexité laisse cependant une large place à l'utilisation de la conservation de la luminance en ue de limiter le nombre de paramètres.

5.1.1 Méthodes fondamentales.

Il existe trois grandes méthodes afin d'estimer localement le flot optique.

- *Les techniques différentielles.* Il s'agit de développer et de dériver l'équation 5.1, puis souvent d'approximer un système du premier ordre afin d'avoir un problème solvable. Cette méthode a l'avantage d'être peu complexe et de permettre une estimation sous-pixellique. L'approximation au premier ordre nécessite néanmoins de respecter les conditions d'un développement limité.
- *Les techniques fréquentielles.* Il est typiquement possible d'utiliser la transformée de Fourier. Soit $F(u_x, u_v, t) = \mathcal{F}[I(x, y, t)]$ la transformée de Fourier 2D de l'image I(x,y,t), en reprenant les notations de l'équation 5.1, on a

$$\mathcal{F}\left[I(x - v_x, y - v_y, t)\right] = F(u_x, u_y, t)\mathrm{e}^{-j2\pi(v_x u_x + v_y u_y)}. \tag{5.2}$$

  Ainsi, le module est invariant par translation comme il a déjà été évoqué au chapitre 3, et la phase $\arg\left(F(u_x, u_y)\right)$ observe

$$\arg\left(\mathcal{F}\left[I(x, y, t)\right]\right) - \arg\left(\mathcal{F}\left[I(x, y, t-1)\right]\right) = -2\pi\left(v_x u_x + v_y u_y\right). \tag{5.3}$$

  Cette dépendance linéaire permet en théorie de résoudre un système linéaire vérifiant cette propriété pour plusieurs fréquences. Cependant cette technique fonctionne bien en pratique pour un objet animé d'un mouvement uniforme sur un fond fixe, mais devient difficile à mettre en place pour des contenus de mouvements plus complexes.
- *L'appariement de fenêtres entre les images.* Cette technique aussi connue sous le terme *bloc matching* a été présentée dans le chapitre 2, puisque c'est la méthode simple et robuste implémentée dans les standards de compression MPEG-2, H.263, MEPG-4/AVC... Afin d'utiliser un maximum d'outils cohérents avec les standards de compression joints, ce type d'algorithmes sera utilisé et développé dans la suite de ce chapitre.

La somme des écarts au carré donnée par

$$E(\Delta_x, \Delta_y) = \frac{1}{N^2} \sum_{(x,y)\in\beta^2} \left(I(x, y, t) - I_{ref}(x - \Delta_x, y - \Delta_y, t - \Delta t)\right)^2 \tag{5.4}$$

est typiquement utilisée entre un bloc courant et un bloc candidat de taille $\beta \times \beta$. Il est à noter que la corrélation évaluée sur ces blocs utilise les hypothèses de conservation et de cohérence (smoothness) du signal entre les deux images.

Il est par ailleurs possible d'augmenter la résolution de l'image le temps de cette étape. Dans le standard H.264, ainsi qu'avec la méthode utilisée pour notre schéma, et décrite dans les sections suivante, une résolution au quart de pixel est atteinte. La figure 5.2 présente les positions au demi pixel, obtenues via un filtrage à 6 coefficients, puis les positions au quart de pixel, issues d'une interpolation bilinéaire.

Une zone de l'image de référence, dans laquelle se fait la recherche d'appariements de blocs, est délimitée pour limiter le coût d'une telle opération. Pour les mouvements rapides, cependant, celle-ci doit être assez grande, ce qui augmente considérablement la complexité de l'algorithme, à l'image de la synthèse de texture à laquelle on aurait augmenté les

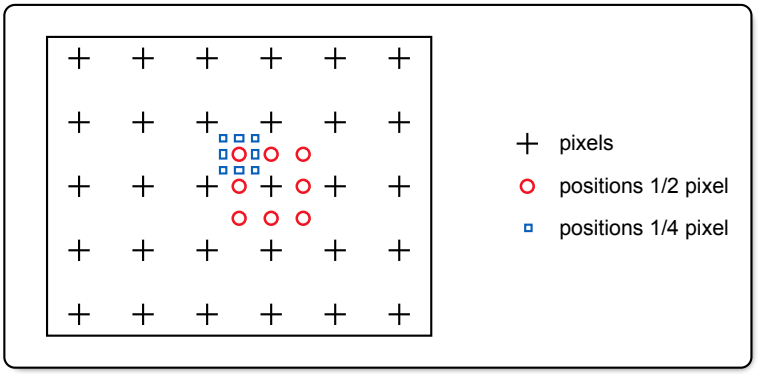


Figure 5.2 – *Précisions atteignables avec l'estimation de mouvement*

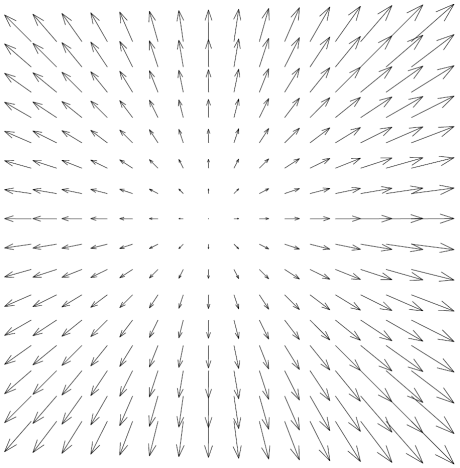


Figure 5.3 – *Champ de vecteurs correspondant à un facteur de zoom*

dimensions du patch source. Pour pallier ce problème, un *estimateur hiérarchique* est communément usité pour permettre, à l'instar de la synthèse multi-résolution de L.Y. Wei et M. Levoy [WL00], de limiter la corrélation entre l'extension de cette fenêtre et la complexité induite. Dans l'approche proposée, un estimateur hiérarchique de mouvement est ainsi utilisé pour déterminer le mouvement en chaque position. Le processus commence par chercher dans l'image du haut de la pyramide (celle de résolution la plus faible), pour raffiner ensuite sa recherche avec les niveaux de résolution supérieure. Le champ de vecteur obtenu au niveau 0 de la pyramide constitue les données de sortie de l'algorithme.

5.1.2 Modèles d'estimation globale de mouvement.

Les méthodes d'appariement dont il est question dans les standards correspondent à une estimation locale du flot optique. Chaque pixel ou bloc possède alors un vecteur de mouvement de translation. Afin de segmenter et modéliser le déplacement des objets d'une scène de la caméra, il est nécessaire d'appréhender une modélisation prenant en compte d'autres types de mouvements, notamment les rotations et le zoom. La figure 5.3, proposée dans [Bru01], illustre le cas d'un champ de vecteurs induit par un zoom. Dans le cas d'une estimation locale, la modélisation nécessite la transmission de $N \times M$ vecteurs, suivant la taille de l'image et de la densité de vecteurs choisie. Or, si l'on arrive à décrire le zoom de manière paramétrique, seules les informations du centre et de l'intensité sont

nécessaires pour décrire le champ en tout point. Malgré le fait que l'on trouve rarement un cas idéal comme celui-ci dans les séquences naturelles, les modèles assurant d'autres types de mouvements que la simple translation permettent d'approximer plus efficacement les mouvements des objets.

Parmi ces modèles on peut citer :

- *Le zoom* dont chaque vecteur de mouvement à la position (x, y) est donné par :

$$\vec{v}(x,y) = \begin{pmatrix} a_0 + \alpha x \\ b_0 + \alpha y \end{pmatrix} \tag{5.5}$$

  L'ensemble des paramètres θ à déterminer est alors : $\theta_{zoom} = (a_0, b_0, \alpha)$.
- *La perspective* :

$$\vec{v}(x,y) = \begin{pmatrix} \frac{a_0+a_1x+a_2y}{c_0+c_1x+c_2y} \\ \frac{b_0+b_1y+b_2y}{c_0+c_1x+c_2y} \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix} \tag{5.6}$$

  définie par les paramètres : $\theta_{persp} = (a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2)$.
- *Le modèle affine* :

$$\vec{v}(x,y) = \begin{pmatrix} a_0 + a_1x + a_2y \\ b_0 + b_1y + b_2y \end{pmatrix} \tag{5.7}$$

  défini par les paramètres : $\theta_{aff} = (a_0, a_1, a_2, b_0, b_1, b_2)$

Ce modèle permet de décrire les mouvements du plan associés à des champs affines. Ainsi, les paramètres de θ_{aff} décrivent les translations, rotations et facteurs de zoom du support utilisé : un bloc, une région ou l'image entière. En proposant un bon compromis entre la fidélité de la modélisation et la complexité des modèles polynômiaux d'ordres supérieurs, la méthode affine a été choisie. Le paragraphe suivant présente la manière utilisée pour résoudre le système induit afin de déterminer les valeurs des 6 paramètres de θ.

Résolution du problème.

Afin de déterminer les paramètres du modèle affine, la régression par la méthode des moindres carrés est utilisée. Soit r_i le résidu associé à la i^{eme} donnée d_i. Il correspond à la différence entre la i^{eme} observation et sa valeur obtenue par le modèle $M(i; \theta_{aff})$, tel que

$$r_i = d_i - M(i, \theta_{aff}). \tag{5.8}$$

La méthode des moindres carrés vise alors à trouver les paramètres de θ_{aff} qui minimisent l'énergie E correspondant au carré des résidus, soit

$$E = \min_{\theta} \sum r_i^2. \tag{5.9}$$

Dans le cas de notre modèle affine, l'opération consiste alors à minimiser

$$E = \min_{\theta} \sum_{i=1}^{N} \Big((d_i - a_0 - a_1x_i - a_2y_i)^2 + (d_j - b_0 - b_1x_i - b_2y_i)^2 \Big) \tag{5.10}$$

Bien évidemment, un tel modèle ne permet pas de décrire à lui seul les mouvements contenus dans les images, puisqu'il décrit le mouvement d'un support cohérent : il faut donc segmenter les images en régions cohérentes en termes de mouvement. La figure 5.4 présente donc, dans ce cas, la chaîne adoptée afin de segmenter les régions sur lesquelles seront

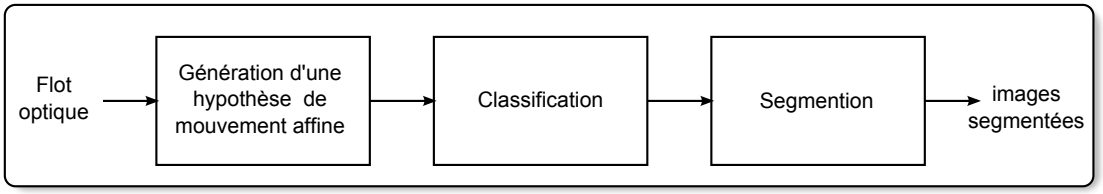


Figure 5.4 – *Processus de segmentation des régions au sens du mouvement*

appliquées les modèles de mouvement déterminés. Néanmoins, le standard MPEG-4 partie 2 contient un outil d'estimation globale, permettant déjà de compenser le mouvement de la caméra, lorsque l'estimation est jugée satisfaisante.

La section suivante décrit le schéma global contenant les étapes citées ci-dessus, et les modifications appliquées pour le codeur et le décodeur.

5.2 Schéma global.

Le schéma du codeur présenté dans le chapitre précédent conduit à la définition de deux types de blocs une fois l'analyse des textures opérées : les blocs de textures dédiés à être synthétisés ainsi que les contours et les patchs qui sont encodés de façon classique. Le schéma temporel offre un troisième mode pour les régions entièrement reconstruites grâces aux informations de mouvement. La figure 5.5 illustre le schéma global. On note premièrement que les images de référence sont entièrement codées classiquement. Elles servent de référence pour la synthèse mais aussi pour l'estimation de mouvement. Pour rappel, ces images de référence ne sont pas à confondre systématiquement avec des images de type I pour H.264. En effet, des images P peuvent servir de référence. Dans ce cas, elles seront classiquement encodées. L'efficacité des modèles affines décroît avec l'éloignement temporel des images de référence. Il n'est donc pas pertinent de faire coïncider références temporelles et images I, ces dernières pouvant être très éloignées suivant les profils utilisés.

Après la segmentation en mouvement permettant de délimiter les régions rigides correctement compensées, le reste des images intermédiaires est soumis à l'étape de segmentation/caractérisation des textures décrite dans le chapitre précédent. Les régions sont alors de même séparées en textures synthétisables et en contours à encoder classiquement. Le train binaire est finalement composé :

- des données relatives aux MB classiquement encodées,
- des paramètres des algorithmes de synthèse pour les régions concernées,
- des paramètres de mouvement des régions à compenser,
- et des informations de localisation des différents MB à interpoler ou synthétiser.

5.3 Segmentation.

Cette section propose de détailler l'utilisation faite des outils d'estimation/compensation de mouvement évoqués en début de chapitre.

5.3.1 Segmentation et setimation de mouvement.

L'estimateur utilisé permet de déterminer les paramètres de mouvement de n régions cohérentes, n étant déterminé a priori. Ce nombre serait de 2 ar exemple pour une scène où un objet se déplace sur un fond fixe. La détermination du nombre de région n'a pas

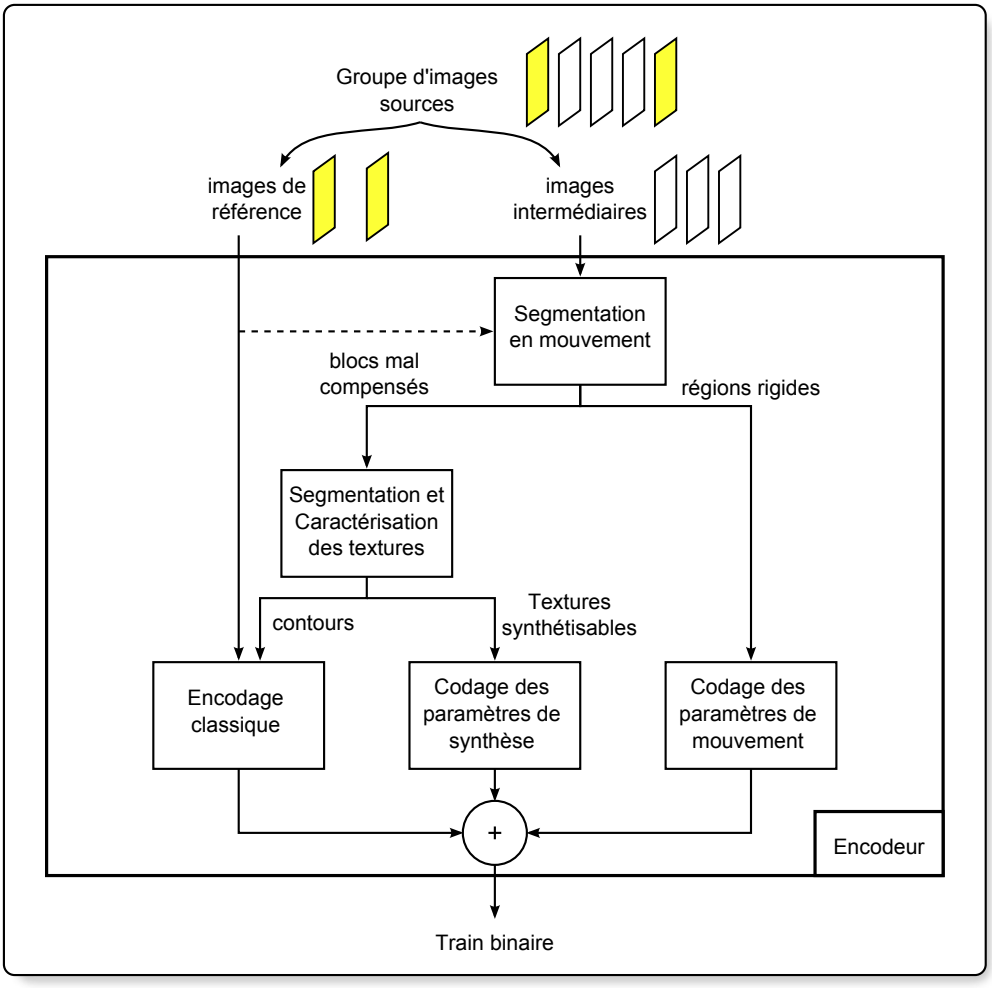


Figure 5.5 – *Schéma bloc du codeur proposé.*

pu être traité, faute de temps, et constitue un des principaux travaux restants à court terme. Après une estimation locale hiérarchique décrite en section 5.1.1, le mouvement prédominant est recherché, *i.e.* le mouvement estimable par les 6 paramètres du modèle affine qui satisfasse le maximum de pixels. Une fois cette étape accomplie, les vecteurs rejetés non cohérents avec les paramètres déterminés, servent à l'itération suivante, afin de déterminer le deuxième mouvement prépondérant. Ce processus itératif se termine quand le nombre de régions prédéterminé est atteint. A la fin de cette étape, tous les pixels de l'image ont un modèle de mouvement associé, même si ce modèle ne leur correspond pas. La figure 5.6 illustre le cas simple du début de la séquence *Coastguard* sur laquelle la caméra suit le bateau au centre de l'image. Dans le cas où le nombre de mouvements cherchés est fixé à 2, les deux régions principales sont séparées par un segment blanc horizontal. En effet, le rivage en arrière plan est animé d'une translation rectiligne uniforme $\vec{v_1}$ et le mouvement global affine déterminé pour le cours d'eau correspond à une translation $\vec{v_2}$. Cet exemple permet de montrer deux failles à cette première segmentation.

- La première réside dans la détermination a priori du nombre de régions à chercher. On s'aperçoit par exemple que le bateau est interpolé avec les paramètres de $\vec{v_2}$ alors que cette séquence aurait nécessité la détermination d'un troisième jeu de

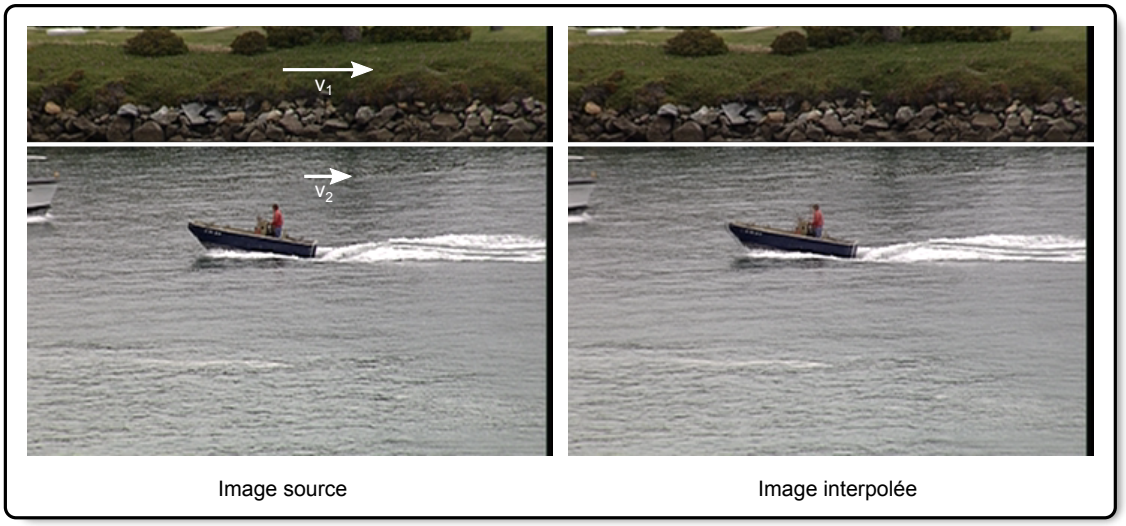


Figure 5.6 – *Interpolation défaillante d'une image de la séquence* Coastguard *CIF. Deux mouvements sont estimés, $\vec{v_1}$ pour la région supérieure au segment blanc, et $\vec{v_2}$ pour la zone inférieure.*

paramètres pour modéliser son mouvement. Cependant, il est périlleux pour ce genre de processus itératif de déterminer un grand nombre de régions. Ce problème sera donc traité par la suite en déterminant si les interpolations sont cohérentes ou non pour chacun des blocs 8×8 ou 16×16 de l'image.

- Les mouvements locaux de la surface de l'eau ne sont pas pris en compte dans le modèle global $\vec{v_2}$, l'interpolation de cette région est donc elle aussi défaillante. Ici aussi, il faut déterminer si l'interpolation produit un résultat satisfaisant, ce qui sera le cas pour les textures et objets rigides, dont les mouvements, capturés en 2D par la caméra, sont descriptibles par un modèle affine.

Pour cela, l'estimateur de mouvement va occasionner une compensation de mouvement dite « forward » entre l'image du milieu du GOP et l'image de référence qui la précède, et une « backward » entre l'image du milieu et l'image de référence suivante. La figure 5.7 illustre le cas d'un GOP comportant 7 images intermédiaires. Ainsi, n régoins sont calculées pour chacune des directions. Il s'agit ensuite de calculer séparément l'interpolation de chaque pixel de l'image centrale à partir de la référence « forward » et du meilleur modèle associé. On fait de même en interpolation « backward ». L'étape suivante consiste à comparer les valeurs interpolées en chaque position afin de déterminer si les modèles correspondent ou si l'estimation est de mauvaise qualité. Une approche par bloc est utilisée : pour chaque bloc est calculée l'erreur

$$\epsilon = \frac{1}{N^2} \sum_{i=0}^{N^2} (I_{fwd}(i) - I_{bwd}(i))^2 \tag{5.11}$$

où N représente la taille du bloc considéré, I_{fwd} et I_{bwd} correspondent aux images interpolées suivant les références *forward* et *backward* respectivement. Si cet écart est inférieur à un seuil déterminé a priori, le bloc considéré se voit assigner les paramètres de mouvements. Sinon, le bloc est considéré comme mal estimé par le meilleur des n régions, il ne sera pas uniquement compensé en mouvement.

La figure 5.8 illustre la détermination des blocs correctement estimés et ceux dont l'interpolation est considérée comme défaillante. Sur la séquence *coastguard*, la caméra

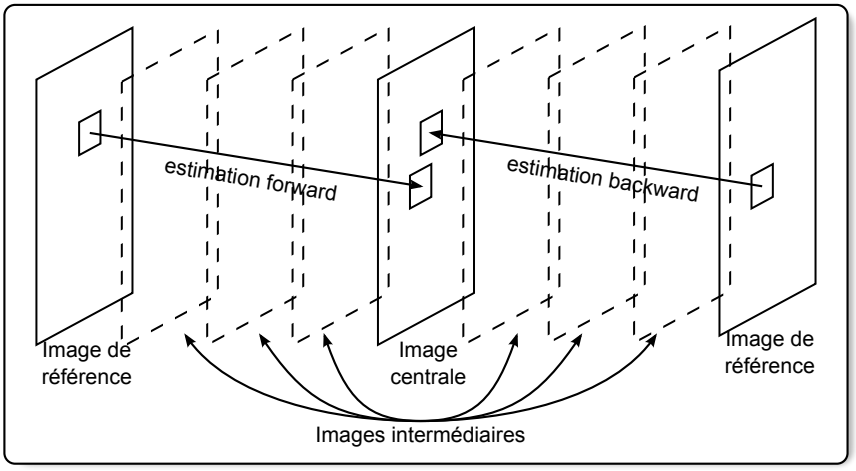


Figure 5.7 – *Estimation des mouvements « forward » et « backward » de l'image centrale du groupe d'images*

suit le bateau des garde-côtes, c'est donc la côte en arrière plan qui semble se déplacer de droite à gauche avec un mouvement rectiligne uniforme. Tous les pixels référant à la côte en arrière plan sont animés du même mouvement par rapport à la caméra. On définit le terme de textures **rigides** ces régions dont le mouvement peut-être estimé avec succès par les paramètres du modèle affine. Cette configuration conduit ainsi à l'étiquetage de la région comme cohérente en mouvement, elle peut ainsi être retirée sur les images intermédiaires. Les images de références décodées ainsi que les paramètres transmis permettront ensuite au décodeur de reconstruire ces régions.

5.3.2 Segmentation spatiale.

La segmentation spatiale reprend celle développée dans le chapitre précédent. Une fois la segmentation déterminée, deux types de blocs se distinguent dans les images intermédiaires des GOP : les blocs compensés en mouvement avec succès et les autres. Dans le deuxième cas, il peut s'agir de blocs de textures qui peuvent être reconstruits pas la synthèse, mais aussi de contours ou de textures non synthétisables par les outils mis en œuvre : ils seront alors classiquement codés en mode intra ou inter image grâce aux images de références.

Il faut ainsi déterminer tout au long du GOP si les blocs considérés contiennent des textures qu'il est possible de reproduire avec les méthodes pixel et patch décrites dans le chapitre précédent et dont l'extension au domaine temporel est présentée dans la suite du chapitre. Cette approche est appliquée image par image : la cohérence des segmentations entre deux images successives n'est donc pas intrinsèquement assurée. Il a ainsi été décidé d'appliquer la méthode de segmentation spatiale sur les images de référence et l'image centrale du GOP. De fait, si le bloc considéré appartient à une région texturée présente sur ces trois images, alors, il est synthétisable.

Le paragraphe suivant décrit l'agencement et la fusion des segmentations temporelle et spatiale.

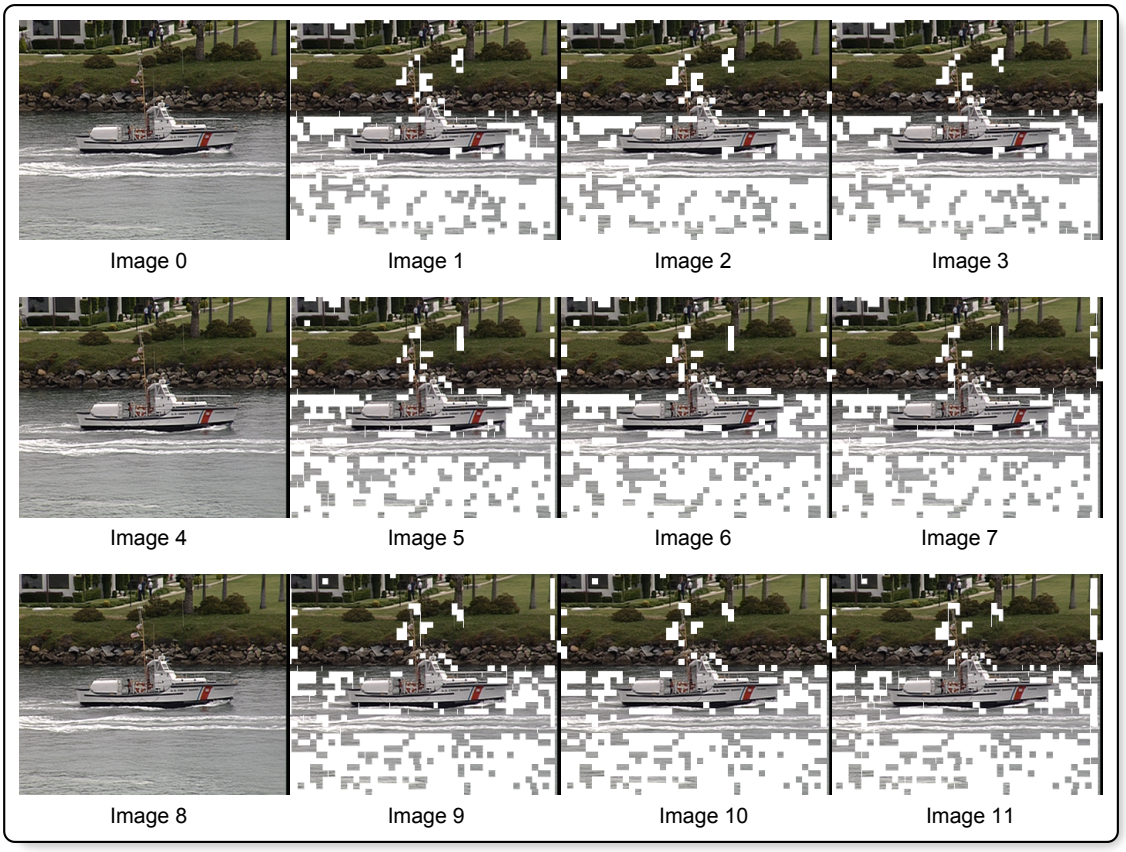


Figure 5.8 – *Segmentation temporelle illustrée pour quelques images de la séquence* Coastguard *CIF. Les images de référence sont séparées d'un intervalle de 4 images. Les blocs blancs correspondent aux zones mal compensées.*

5.3.3 Construction des régions à synthétiser.

La prise en compte d'une séquence d'images et la compensation de mouvement vont permettre de réduire le nombre de blocs à encoder grâce notamment aux régions qui pourront être directement interpolées via les images clés en début et fin de GOP associées aux modèles de mouvement calculés. La figure 5.8 montre qu'il reste souvent un grand nombre de blocs qui ne sont pas correctement interpolés. Parmi ceux-ci, on trouve des blocs contenant uniquement de la texture, au sens de la segmentation spatiale adoptée, et d'autres, contenant des contours. Pour ce dernier type, on considère trivialement que la seule issue consiste à encoder classiquement ces blocs afin qu'ils soient correctement reconstruits par le décodeur. Pour les premiers, par contre, il va falloir déterminer comment construire les régions candidates pour appliquer la synthèse de texture.

Dans le chapitre 4 a été présentée une manière de délimiter des régions texturées avec le patch constitué des MB entourant une zone rectangulaire, potentiellement enrichie de blocs d'ancrages. Une première implémentation de test a donc été logiquement issue de la fusion de ce découpage avec la segmentation temporelle. Ainsi, la figure 5.9 présente la segmentation temporelle et le découpage spatial d'une des images intermédiaires d'un GOP de la séquence CIF *Container*. L'image (d) présente les régions finalement retenues pour la synthèse. Le reste de l'image est alors dédié à être encodé classiquement pour les blocs blancs sur l'image (b) ou interpolé grâce aux modèles de mouvements déterminés. A l'intérieur de ces régions maintenant, l'interpolation des blocs dont le mouvement a été

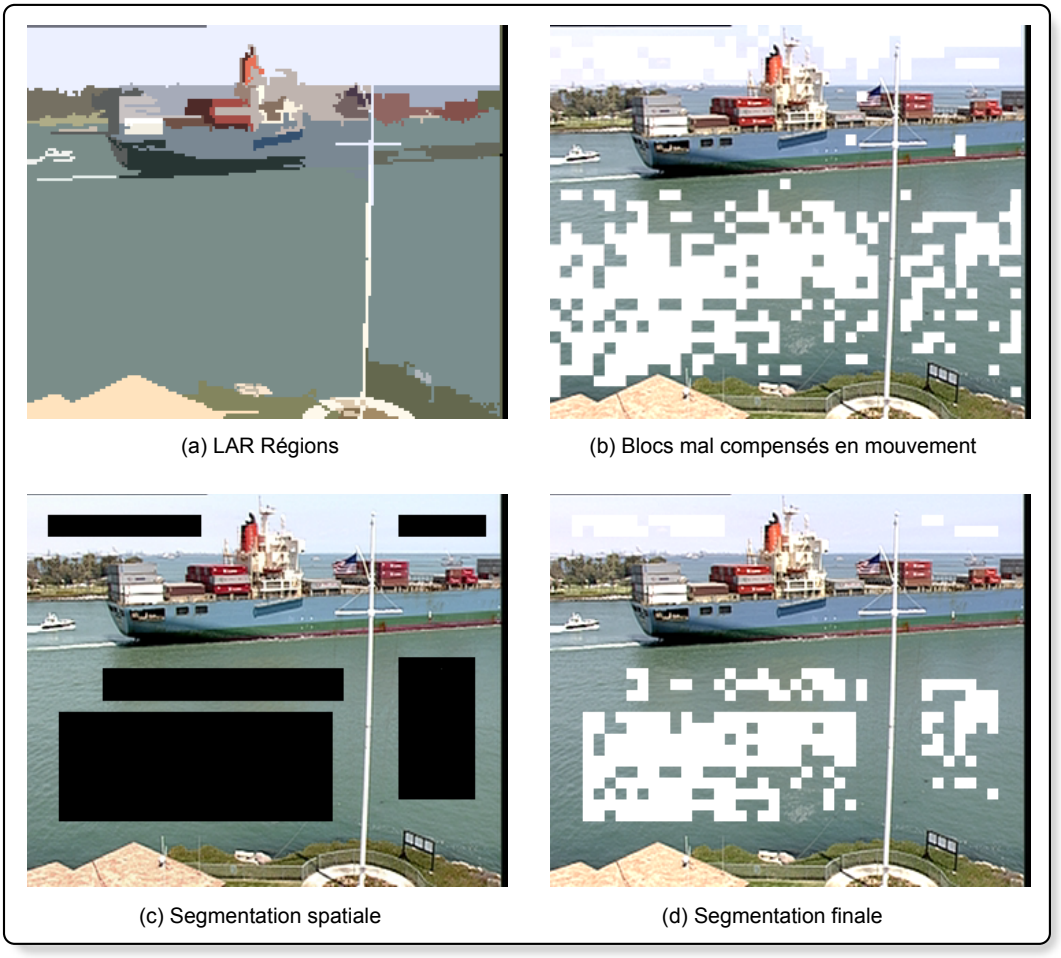


Figure 5.9 – *Première fusion des segmentations spatiale et temporelle.*

correctement estimé, est conservée afin de servir de blocs d'ancrage. Aussi, aucun bloc d'ancrage n'a été déterminé au moment de la segmentation spatiale, ceci afin d'éliminer le codage de ces blocs coûteux, difficilement prédictibles spatialement puisque dépourvus de contexte spatial sur lequel fonder une prédiction spatiale. De la même manière que pour le découpage des régions présenté au chapitre précédent, des blocs d'ancrage seront donc codés uniquement dans le cas où la région excède un nombre déterminé de MB en longueur et largeur, mais aussi à la condition qu'aucun bloc correctement interpolé ne soit capable d'assurer ce rôle.

On s'aperçoit cependant que cette méthode de découpage laisse peu de place à l'impact de la synthèse de texture. En effet, de nombreux blocs texturés, en dehors des régions segmentées, devront être classiquement encodés. Ce découpage strict répondait aux contraintes d'un patch construit dans la même image, autour des régions retirées. Or, dans le cas du traitement d'un groupe d'images, il est possible de construire un patch dans les images de références. Ainsi, les contraintes de contour sur les régions à synthétiser pourraient être en partie levées.

Pour résumer, la figure 5.10 illustre les différentes étapes, menant à la construction de la segmentation finale, fondée sur l'étude des mouvements et des textures contenus dans un GOP source.

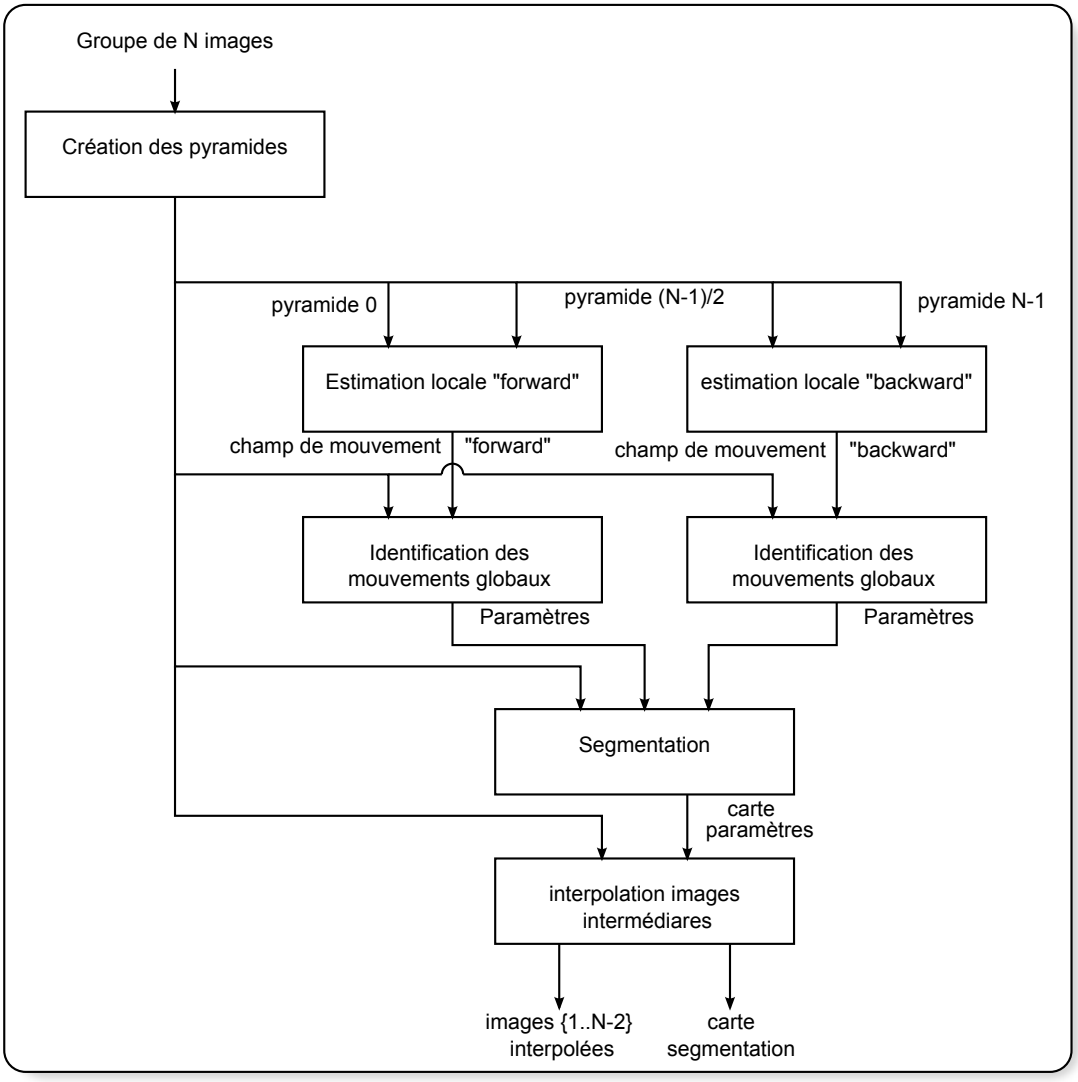


Figure 5.10 – *Schéma d'estimation, de segmentation et de compensation de mouvement pour un GOP de N images*

5.4 Synthèse temporelle.

Cette section présente les différentes synthèses adoptées suivant les types de textures rencontrées. Comme pour la synthèse du modèle intra, les synthétiseurs basés pixel et patch seront utilisés et adaptés au domaine temporel. Pour les textures segmentées par le module d'analyse du mouvement comme régions rigides, dont le mouvement a été paramétré avec succès, la synthèse est inutile étant donné que la transmission d'un minimum de paramètres permet de les reconstruire au décodeur.

Il est ici nécessaire de définir les patchs de texture adoptés exploitant les images précédemment reconstruites. De même, il est utile d'adapter les synthèses aussi à l'estimation/compensation de mouvement à disposition ainsi qu'à l'ordre de traitement des images des GOP par le système d'encodage joint.

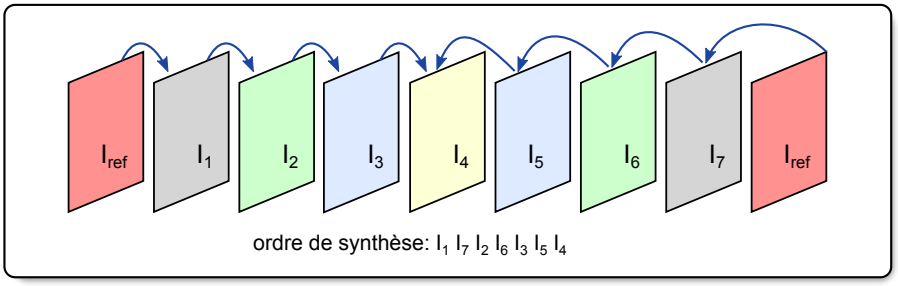


Figure 5.11 – *Ordre de synthèse des images d'un GOP fondé sur la confiance.*

5.4.1 Synthèse dans un GOPM.

La première évolution majeure réside dans le fait que les images de référence peuvent fournir les patchs d'entrée pour la synthèse. Ainsi, les difficultés de recherche de compromis entre taille de patch et taille des régions enlevées, apportant les gains en débit, sont en partie levées. Les régions enlevées possèdent maintenant des patchs colocalisés dans les images de référence.

Aussi, il paraît nécessaire de définir un ordre temporel de synthèse des images à la manière de l'ordre spatial de reconstruction des régions. Dans ce cadre, plusieurs solutions ont été testées, avec pour chacune ses avantages et ses inconvénients.

Une première méthode envisagée est inspirée de l'ordre fondé sur une carte de confiance, développé dans le chapitre 4. La figure 5.11 illustre l'ordre dans lequel sont traitées les images. Ainsi, le voisinage temporel des images est considéré en termes de confiance. Les images de référence sont dotées d'une confiance maximale, les premières images synthétisées sont donc en premier lieu les images 1 et 7. La synthèse est ensuite appliquée sur les images de proche en proche. Cette méthode présente l'avantage de commencer la synthèse d'images possédant des références et donc des patchs très proches en termes de textures. Cependant, comme c'est le cas pour l'ordre spatial (*cf.* les artefacts de frontières conduisant à la création des blocs d'ancrage), l'image centrale va concentrer les artefacts visuels issus des divergences progressives de proche en proche. Un décalage gênant pour l'observateur intervient donc au milieu de chaque GOP. Pour tenter de palier ce problème en conservant un ordre fondé sur la confiance, une remise en cause des images synthétisées parait inévitable. Toutefois, le coût calculatoire engendré s'avère tout à fait prohibitif.

La méthode retenue consiste donc à synthétiser les images suivant un modèle hiérarchique tel qu'on le trouve dans le standard scalable SVC (Scalable Video Coding)[SMW07]. La structure du groupe d'images est alors hiérarchisée comme illustré sur la figure 5.12. Cette dernière décrit deux cas de figures en fonction de la structure utilisée par le codeur joint. En effet, toujours dans l'optique de préserver la cohérence entre les outils de compression classique et la synthèse, l'ordre du processus doit permettre de s'adapter et ainsi de synthétiser en premier les images les plus pertinentes. Dans un GOP de type H.264, les deux options suivantes peuvent être envisagées.

- Synthétiser uniquement les images de type B. Les images I et P servent alors d'images de référence. La méthode est alors compatible avec un schéma comportant des images B hiérarchiques, introduites dans la section 2.4.2.
- Synthétiser les images B et certaines P. Seules les images I et les P entièrement encodées sont alors disponibles comme références. Cette approche, plus ambitieuse, préserve alors le débit de régions présentes dans les images P qui nécessitent l'allocation d'un débit supérieur aux images bi-prédites. Il faudra néanmoins surveiller la

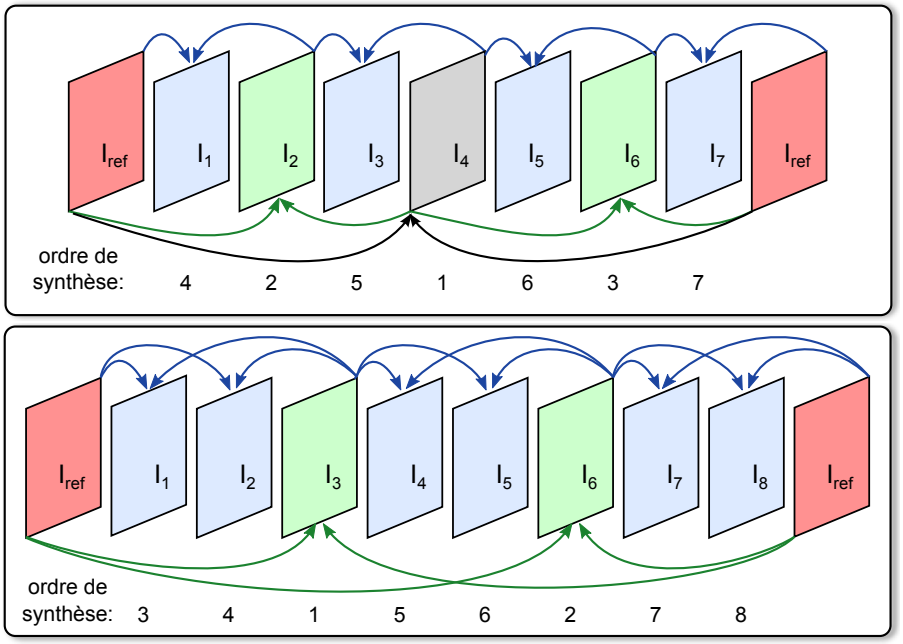


Figure 5.12 – *Ordre hiérarchique de synthèse des images d'un GOP.*

qualité des synthèses des images dont les patchs de référence se trouvent temporellement éloignés.

Les deux paragraphes suivants exposent l'adaptation des deux algorithmes de synthèse afin de préserver un maximum de cohérence entre les images.

5.4.2 Adaptation des synthèses *pixel* et *patch*.

La synthèse de texture, dans ce contexte, est utilisée pour les régions dont les mouvements locaux n'ont pas permis une modélisation affine globale sur la zone. C'est le cas par exemple sur les surfaces aquatiques des séquences *Coastguard* et *Container*. Ces régions ont une place importante en termes de surface de l'image. Cependant, les mouvements locaux de l'eau sont détectés via l'étape de segmentation temporelle. Le plan d'eau possède néanmoins un mouvement global dont s'écartent plus ou moins les mouvements locaux (les vagues). Un guide a alors été ajouté afin d'aider à la création d'une cohérence temporelle acceptable entre les images successives de la séquence. Celui-ci est constitué de l'image compensée en mouvement à partir des paramètres de mouvement. Elle contient, contrairement aux images présentées sur la figure 5.8, les pixels interpolés évalués comme défaillants (en blanc) par l'étape de segmentation. Ces régions apparaissent alors floues sur la figure 5.6, mais elles peuvent néanmoins servir de guide à la synthèse, comme il a été évoqué dans le chapitre 1. La méthode de guidage, décrite dans la suite, diffère cependant de l'approche proposée par M. Ashikhmin dans [Ash01].

L'adaptation des méthodes patch et pixel a ainsi pour but d'éviter les incohérences temporelles grâce au guide issu de la compensation en mouvement. Le traitement de groupe d'images permet d'utiliser comme patchs sources des versions colocalisées de la région à synthétiser. Afin de définir ces images, le terme *image de référence* prend pour cette partie un nouveau sens. Localiser les patchs sources uniquement dans les images I peut provoquer des artefacts de scintillement entre les images successives puisque les pixels recopiés proviennent d'images temporellement éloignées. Aussi, pour une image courante,

si les régions colocalisées des images voisines sont disponibles, elles sont alors considérées comme patch. Les images de référence sont donc les images disponibles **les plus proches**.

L'utilisation de ce guide est illustrée sur la figure 5.13. Pour la synthèse sont ainsi pris en compte :

- v_c correspondant au voisinage courant du pixel ou patch à synthétiser,
- $v_{ref}(1)$ et $v_{ref}(2)$ sont deux des voisinages candidats appartenant aux images de référence $I_{ref}(1)$ et $I_{ref}(2)$ respectivement,
- c_{int} correspond au voisinage colocalisé à v_c appartenant à l'image interpolée à partir de $I_{ref}(1)$ et $I_{ref}(2)$ et des informations de mouvement.

L'écart ϵ calculé pour rechercher, parmi les régions de $I_{ref}(1)$ et $I_{ref}(2)$, le meilleur voisinage s'écrit alors

$$\epsilon = \|v_c - v_{ref}(i)\|_2 + \|v_{int} - v_{ref}(i)\|_2 . \tag{5.12}$$

La norme $\|.\|_2$ correspondant à la somme des écarts des pixels comparés est de nouveau adoptée pour la version temporelle.

Pour le moment, cette construction conduit à la synthèse de région à partir de patchs de grande taille, ce qui induit une synthèse coûteuse en temps de calcul. Une version ultérieure devra permettre de localiser des patchs plus restreints mais pertinents dans les images de référence.

La section suivante présente des résultats préliminaires du schéma de synthèse temporelle. De futurs tests subjectifs permettront d'évaluer plus précisément l'approche proposée.

5.5 Intégration du schéma avec le standard H.264.

5.5.1 Segmentation temporelle implémentée.

Les figures 5.14 et 5.15 illustrent les synthèses basées patch et pixel respectivement sur les images successives d'un GOP de 5 images de la séquence Coastguard, encodée avec H.264 paramétré avec $QP = 10$ et une structure IPPP. Une région a été découpée à la main afin de focaliser l'attention sur la synthèse.

On note sur les figures 5.14 et 5.15 que les blocs correctement estimés à « l'intérieur » de la région à synthétiser sont présents et servent de blocs d'ancrage pour la synthèse. Le décodeur, afin d'établir les images intermédiaires en entrée du synthétiseur doit disposer des images de références, de paramètres de mouvement des régions, mais aussi des positions de ces blocs stratégiques. On s'aperçoit sur les séquences illustrées que ces blocs ne sont pas toujours alignés sur une grille (8×8 ou 16×16 typiquement, selon la taille des blocs). Cependant, la segmentation est appliquée sur l'image centrale du GOP (l'image précédente dans le cas où le GOP contient un nombre pair d'images), sur cette image uniquement. Tous ces blocs sont alors positionnés sur la grille des blocs (8×8 ou 16×16), par la construction de la segmentation développée en section 5.3.1. Avant encodage, une carte est transmise afin de localiser les blocs déterminés sur cette image. Les paramètres de mouvement permettent ensuite de propager ces blocs pour le reste des images intermédiaires.

5.5.2 Le mode *Skip*.

Toute l'approche est fondée sur la réduction du débit des zones synthétisées. Le codeur et l'outil de compensation de mouvement par région sont utilisés conjointement en établissant que les images de référence pour l'estimation correspondent aux images clés du codeur : les images I typiquement, voire les images P, pour l'encodage de longs GOP. Aussi

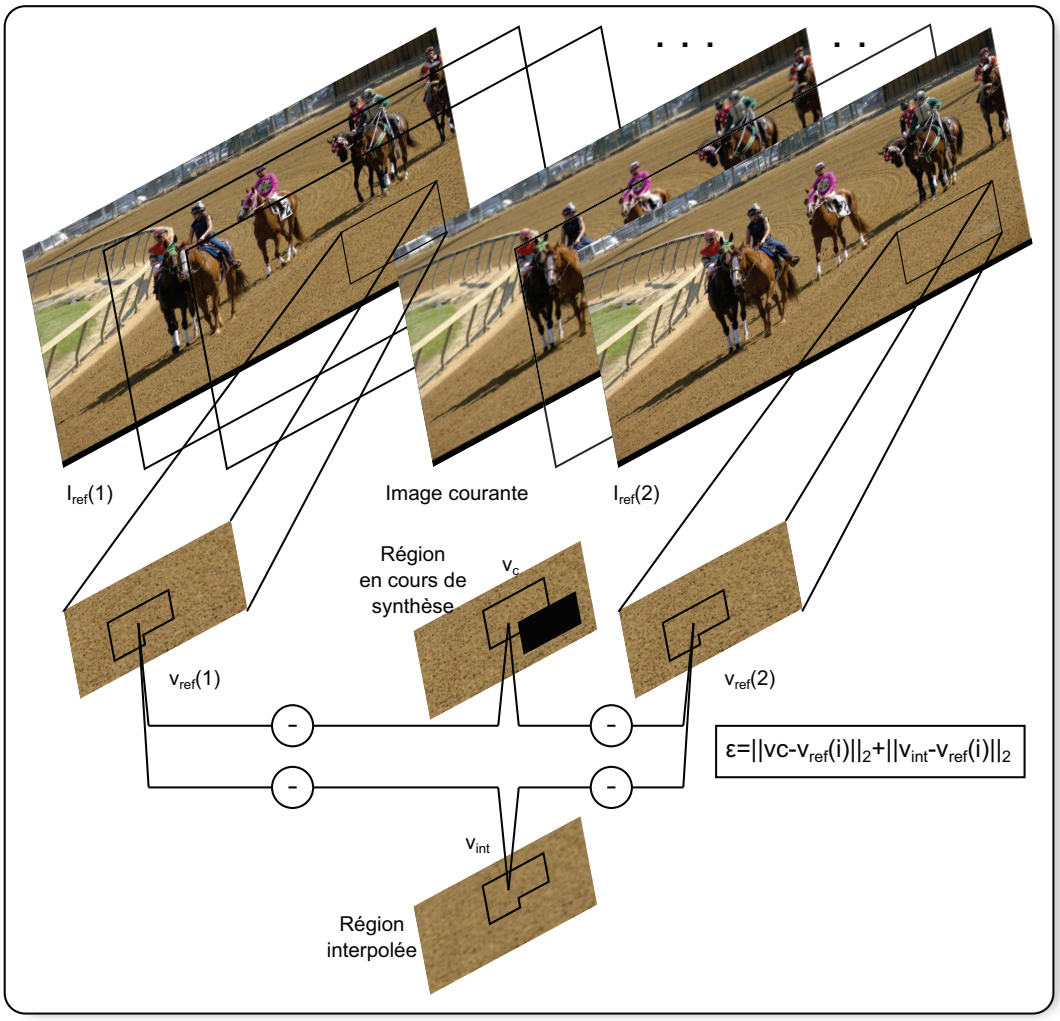


Figure 5.13 – *Adaptation des algorithmes de recherche du meilleur voisinage ou de la meilleure zone de chevauchement au domaine temporel.*

les images synthétisées correspondent à des images de types P et B supportant le mode *skip*. Contrairement au schéma intra présenté dans le chapitre précédent, il faut cependant forcer directement le codeur à utiliser ce mode. De plus, il est nécessaire de s'assurer que le décodeur puisse dissocier les MB à synthétiser des MB *naturellement skippés i.e.* ceux qui n'ont pas été déterminés par l'analyse des textures mais par la décision du mode au codeur. À cette fin, une carte de ces MB forcés au mode skip doit être transmise au décodeur. Heureusement, la quantité d'informations à transmettre réellement pour cette carte est très faible. Premièrement, elle ne concerne que les MB skippés, tous les autres MB seront décodés naturellement. De plus, une seule carte par GOP est nécessaire puisque leur mouvement est connu : seule la carte les localisant dans l'image centrale du GOP est donc nécessaire. Enfin l'entropie est elle aussi faible puisque le mode *skip forcé* concerne souvent de larges régions par construction. Cette information s'ajoute alors aux 6 paramètres des n mouvements.

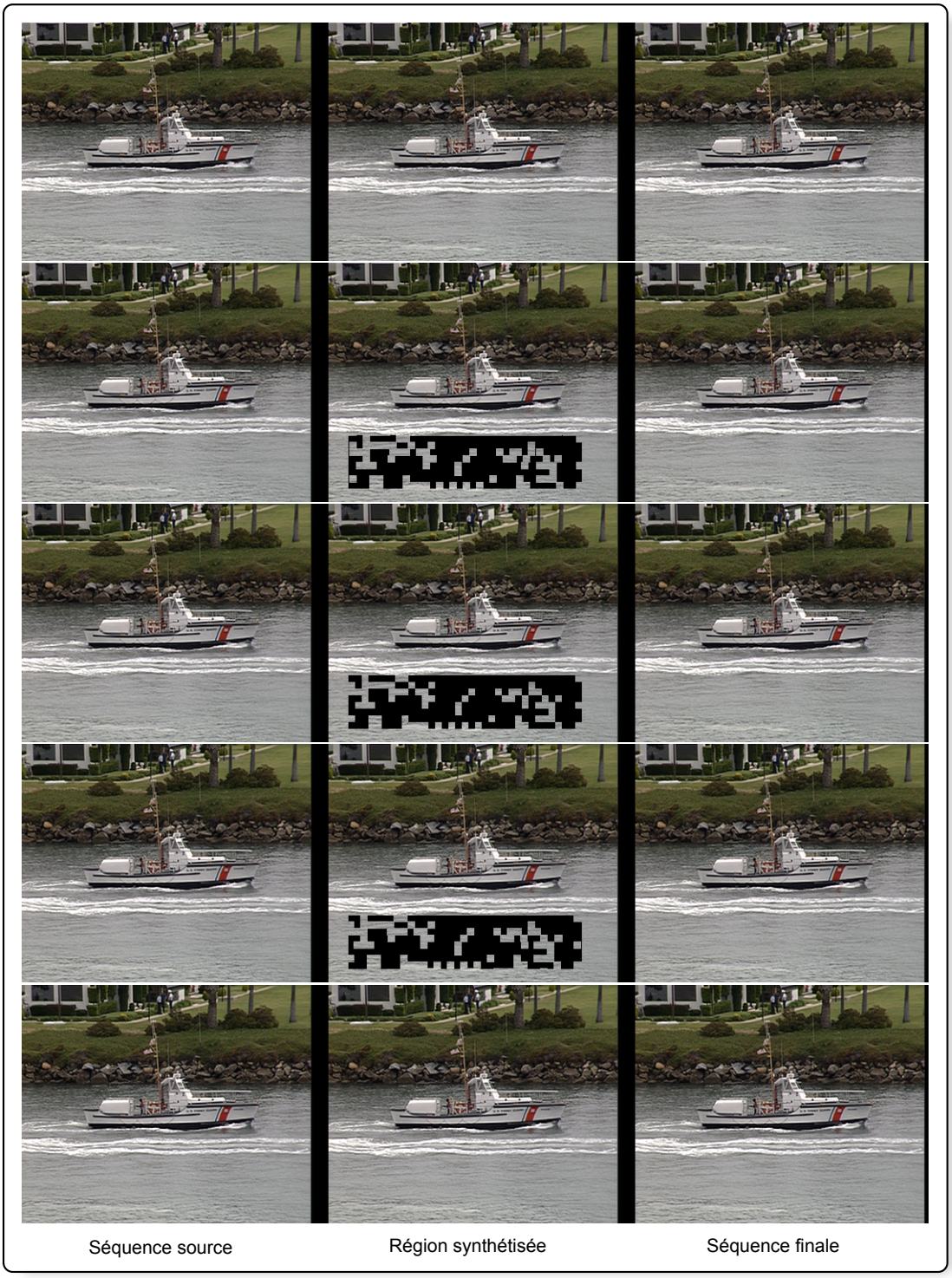


Figure 5.14 – *GOP de 5 images synthétisées par la méthode* patch *des Graphcuts.*

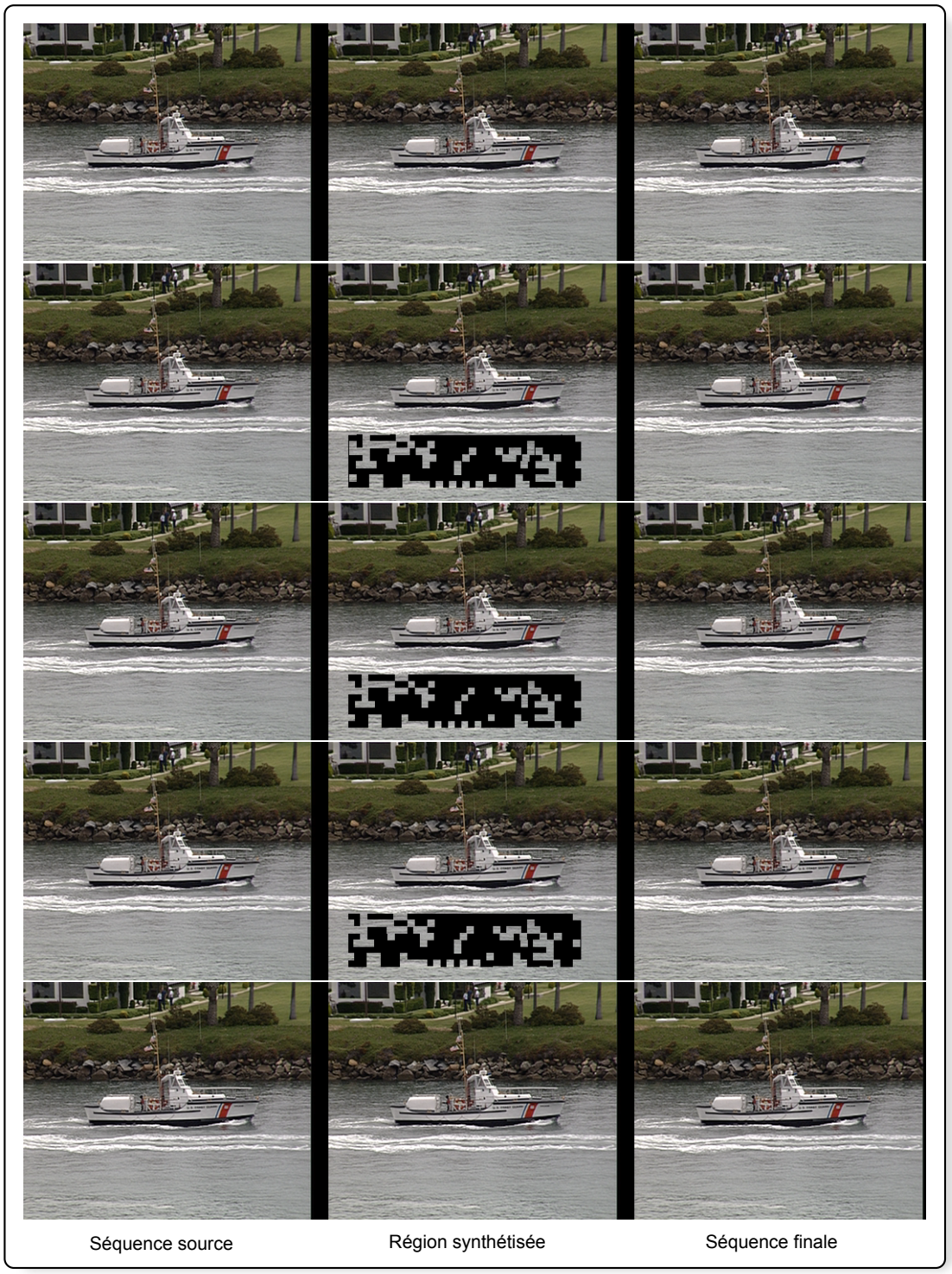


Figure 5.15 – *GOP de 5 images synthétisées par la méthode* pixel.

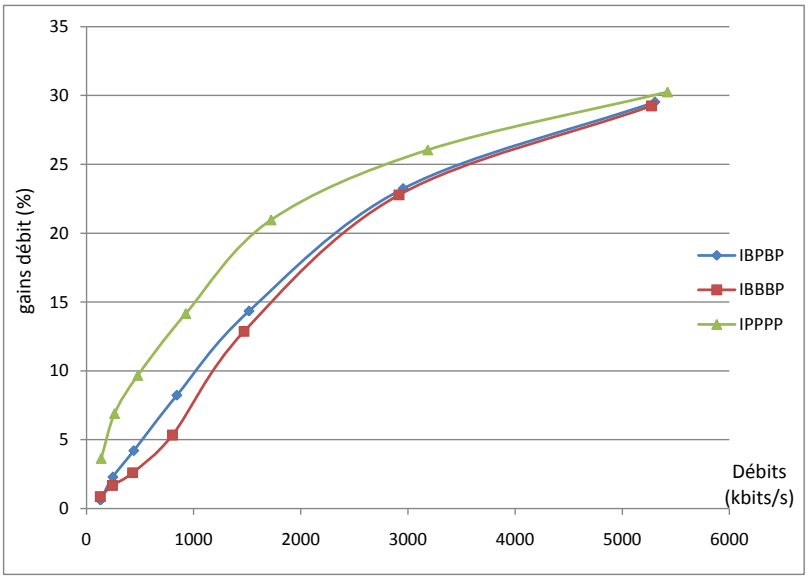


Figure 5.16 – *Gains observés, vis à vis d'H.264, pour la séquence CIF* Container *suivant plusieurs structures de GOP.*

5.5.3 GOP et estimation affine de mouvement.

Plusieurs structures de GOP ont été considérées pour évaluer les résultats visuels et de débit obtenus. La figure 5.16 présente l'évolution des gains en débit observés en fonction des structures de GOP et GOPM utilisées. Empiriquement, on relève sur la segmentation d'une multitude de séquences CIF et SD que l'efficacité de l'estimation affine décroit sensiblement au delà d'un GOPM de 5 images. La figure présente les résultats des gains en débit pour l'encodage de la séquence CIF *Container*, avec les trois structures suivantes, dans lesquelles les images de référence pour le mouvement sont notées en gras :

- **I**PPP**P**PPP**I**...,
- **I**BPB**P**BPB**I**...,
- **I**BBB**P**BBB**I**....

NB : La prédiction des MB des images P est faite à partir de deux images clés. Ainsi, l'image de référence P, dans le deuxième type de GOP peut être prédite à partir de l'image I précédente dans le cas où la prédiction conduit aux MB *skippés* de l'image P précédente. Aussi, le GOP IBBBP admet une structure hiérarchique : les images $I_0B_1B_2B_3P_4$ sont décodées dans l'ordre $I_0P_4B_2B_1B_3$.

On note sur cette figure que la structure IPPP permet des gains supérieurs. En effet, ces derniers sont directement liés au nombre de MB skippés, et au type d'images synthétisées. Les gains sont logiquement moindres dans le cas d'images B, généralement plus efficacement prédites que les images de type P. Cependant, cette figure présente les gains en débit par rapport à H.264 qui est usuellement paramétré pour des structure contenant des images B. Il est donc légitime de se comparer aux autres types de GOP. La figure

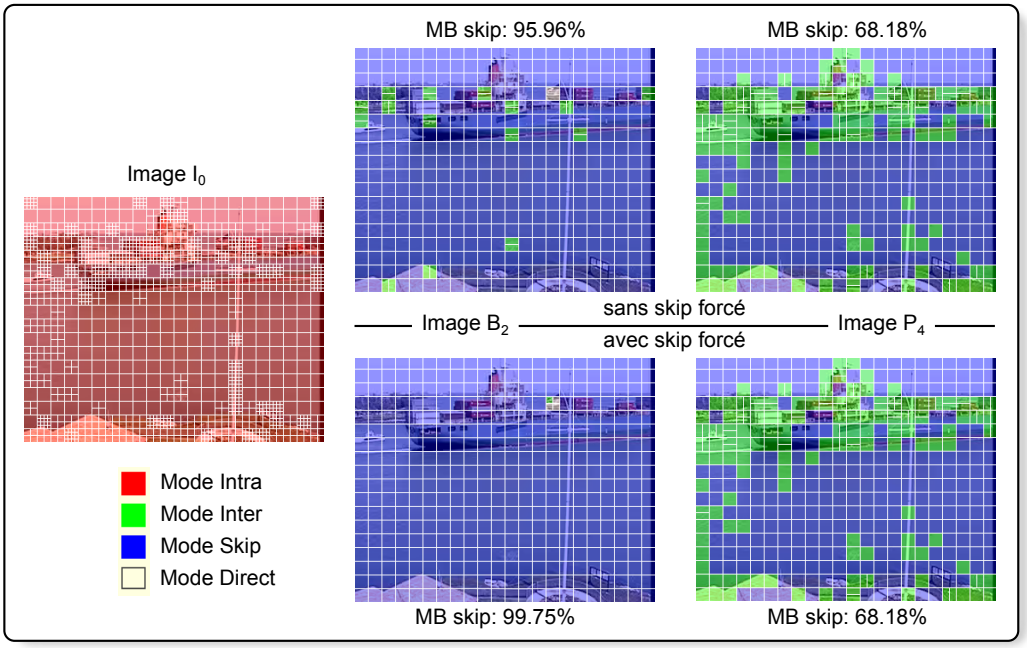


Figure 5.17 – *Cartographie des modes de prédiction choisis pour la séquence CIF* Container. *La structure de GOP suit $IB_1B_2B_3P$, seules les images I, B_1 et B_2 sont représentées.*

5.17 cartographie, pour la séquence *Container*, les modes de prédiction des MB. La structure IBBBP est présentée, avec les images I et P comme images de référence. La ligne du bas présente les modes choisis au codeur avec les skip forcés en fonction de l'analyse des mouvements et textures alors que la ligne du haut correspond à l'encodage classique. Les images de référence sont logiquement identiques puisqu'elles sont encodées en premier et sans skip forcé.

Les gains présentés dans cette section tiennent compte d'une majoration du débit nécessaire pour transmettre les différents paramètres en information supplémentaire. Pour rappel, les informations suivantes doivent être connues du décodeur.

- Les paramètres des n mouvements affines estimés. On ne relève lors de la segmentation que les 3 ou 4 principales régions cohérentes, ce qui donne ainsi 18 à 24 coefficients flottants. dans cette implémentation, les paramètres sont codés sur 32 bits, cette information correspond donc dans le pire des cas à 768 bits par GOPM, ce qui est négligeable devant le débit total d'un GOPM.
- L'information relative au skip naturel ou forcé. Cette information nécessite donc au plus 1 bit par bloc « skippé ».
- enfin, 2 bits par MB skippé, si le nombre de régions a été fixé à 3 ou 4, sont nécessaires pour indiquer son appartenance à une région estimée en mouvement.

Par exemple, pour la séquence CIF *Coastguard* avec 3 régions segmentées en mouvement, 1 kbit par image a été déduit. Cette approximation majore grossièrement cette quantité d'information, une étude approfondie de l'entropie de tels paramètres permettra de la quantifier plus précisément. En effet, les régions étant spatialement cohérentes, un simple codage prédictif permettra de réduire fortement l'information transmise.

Les figures 5.18 et 5.17 permettent d'avoir un aperçu de l'impact des MB skippés sur les modes de codages des autres MB aussi pour une structure IBPBP respectivement IBBBP. Les modes inter, intra, direct et skip sont donnés par la teinte et les tailles des blocs prédits

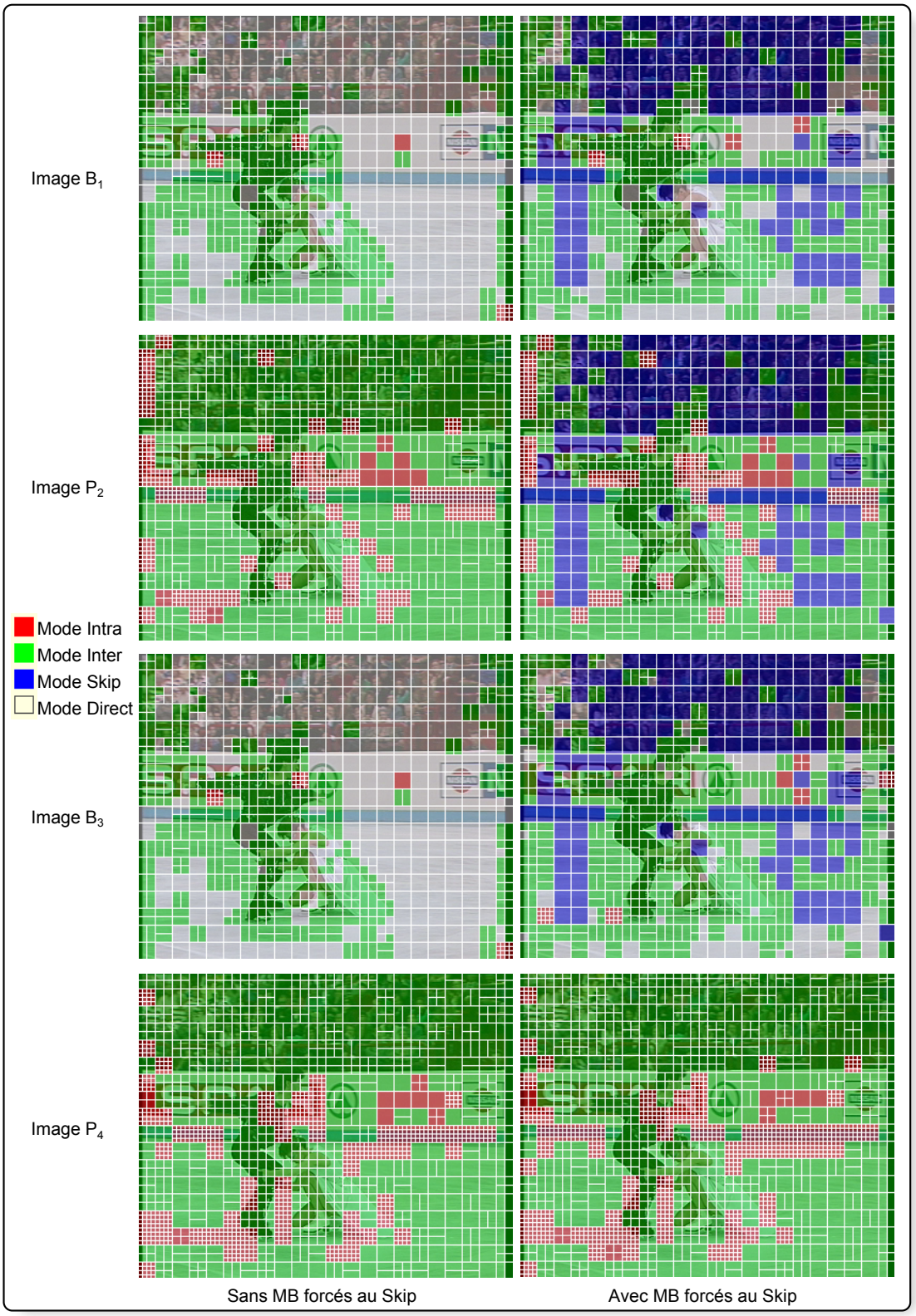


Figure 5.18 – *Cartographie des modes de prédiction choisis pour la séquence CIF* Patinage *avec un* $QP = 15$. *La structure de GOP suit* $IB_1P_2B_3P$, *les images* B_1, P_2, B_3 *et* P_4 *sont représentées.*

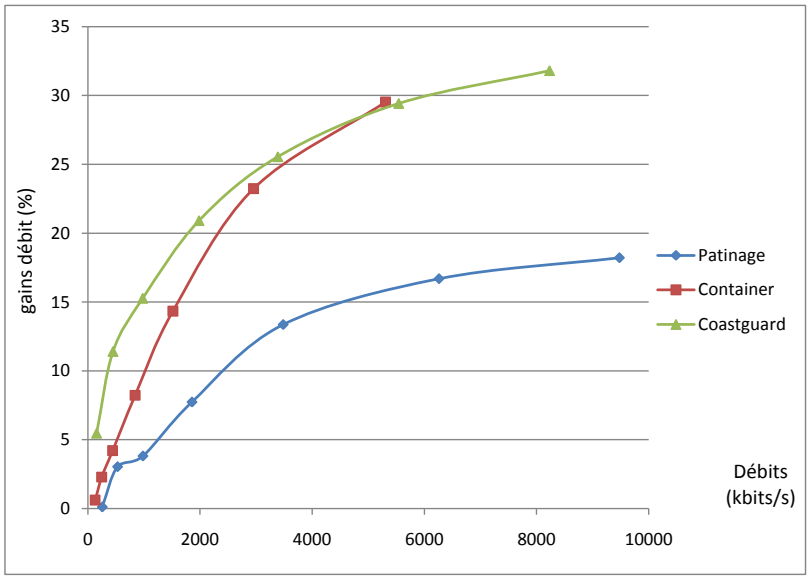


Figure 5.19 – *Comparaison des gains obtenus sur trois séquences CIF :* Coastguard*,* Container *et* Patinage *avec une structure IBPBP.*

sont indiquées par la grille blanche. Le mode direct correspond au mode inter, disponible pour les images B, dans lequel le vecteur de mouvement déterminé correspond au vecteur prédit à partir du voisinage spatio-temporel. Ainsi, il n'est pas nécessaire de transmettre le vecteur de mouvement, seuls les coefficients si non-nuls. Le mode skip pour les images B correspond alors à un MB direct 16×16 avec les coefficients résiduels nuls. On observe sur la figure 5.18 que la deuxième image P n'est pas la même dans les deux cas. Ceci provient du fait qu'elle est prédite à partir de l'image P précédente et de l'image I, l'ordre de codage étant $IP_2B_1P_4B_3$. L'image P étant « perforée » de MB forcés au mode skip, la dernière image ne peut être prédite aussi efficacement. Le cas de la structure IBBBP ne présente pas ce problème étant donné l'ordre de traitement $I_0P_4B_2B_1B_3$.

La figure 5.19 permet de rendre compte de la variation des gains obtenus sur 3 séquences CIF. On observe, au vu des séquences, que les gains dépendent fortement du taux de MB skippés dans les images intermédiaires, mais aussi du contenu corrélé à l'efficacité originale de H.264.

Comme pour le schéma intra présenté au chapitre précédent, ces gains sont issus de l'hypothèse que les images reconstruites sont d'une qualité visuellement comparable à celle encodée sans MB forcés au mode *skip*. La méthode n'étant pas directement évaluable par une métohde objective, la section suivante présente une validation des résultats par une évaluation subjective.

ressenti	note
Imperceptibles	5
Perceptibles, non gênantes	4
Légèrement gênantes	3
Gênantes	2
Très gênantes	1

Table 5.1 – *Ressenti des observateurs sur les dégradations occasionnées.*

5.6 Résultats et évaluation subjective.

Afin de valider l'hypothèse forte que les vidéos reconstruites avec synthèse sont visuellement similaires aux vidéos entièrement décodées de manière classique, des tests subjectifs ont été menés comme pour l'approche *intra* présentée au chapitre précédent. Comme pour l'évaluation des images, une méthode à simple stimulus a été choisie, c'est à dire que les séquences vidéos sont jouées une par une et non côte à côte pour la comparaison. La méthode est néanmoins sensiblement différente puisqu'il s'agit de montrer des séquences vidéo. Ainsi, les observateurs n'ont plus les commandes pour choisir de naviguer entre les séquences, un ordre est imposé. Pour valider rapidement les différentes techniques, un échantillon de 3 séquences CIF est proposé. Il est constitué des séquences Coastguard, Container et Patinage, illustrées en annexe. Pour chacune de ces séquences, 3 niveaux globaux de qualité au sens de la quantification du standard H.264 sont montrés. Ils correspondent à l'encodage avec les paramètres : $QP = 15, 25, 35$ permettant d'évaluer l'approche sur une large gamme de débits. Ensuite, pour chacun de ces niveaux, plusieurs types de traitements sont testés afin de les comparer. Prenons l'exemple du niveau $QP = 25$.

1. La vidéo entièrement décodée, transmise avec un débit D_1.
2. La vidéo synthétisée, transmise avec un débit D_2. On a $D_2 < D_1$ puisque des blocs ont été *skippés* dans le second cas.
3. La vidéo entièrement décodée, transmise avec un débit $D_3 \approx D_2$. Cette vidéo permet de déterminer si la version synthétisée produit au moins un résultat visuel meilleur ou égal à celui produit par H.264 à débit équivalent.
4. La vidéo synthétisée avec l'algorithme intra image, transmise avec un débit D_2. Cette séquence a pour but de valider l'apport de l'introduction de la cohérence temporelle.

L'expérience, à laquelle 17 observateurs ont participé, est calibrée pour durer 10 à 15 minutes par observateur. Les séquences durent 8 à 10 secondes, elles sont présentées deux fois pour chacun des traitements. Tous les traitements n'ont pas été testés pour toutes les séquences et tous les niveaux de qualité, Cette lacune est motivée par la volonté de réaliser un test compact sur la durée pour chacun des observateurs, facteur jugé important pour l'attention portée à chaque séquence, par les observateurs.

La question posée aux observateur porte sur le ressenti des observateurs vis-à-vis des dégradations engendrées par le codage et la synthèse. Le tableau 5.1 décrit l'échelle utilisée pour guider les observateurs dans leurs notations.

Le tableau 5.2 présente les notes moyennes obtenues lors des tests pour la comparaison entre la version classiquement décodée (H.264) et celle avec décodage partiel et synthèse (H.264+synthèse). On s'aperçoit clairement que les écarts sont faibles au niveau du ressenti entre les deux versions. La moyenne de ces écarts est de 0.18, ce qui est nettement inférieur

Container

	H.264	H.264+synthèse
QP=15	5	4.93
QP=25	4.8	4.53
QP=35	3.27	2.93

Patinage

	H.264	H.264+synthèse
QP=15	4.87	4.6
QP=25	4.87	4.6
QP=35	4.2	4.2

Coastguard

	H.264	H.264+synthèse
QP=25	4.67	4.67
QP=35	3.33	3.47

Table 5.2 – *Notes moyennes (sur 5) obtenues pour la comparaison entre les séquences synthétisées et celles entièrement décodées classiquement.*

	H.264	H.264+synthèse
$D_{QP\approx 15}$	4.87	4.93
$D_{QP\approx 25}$	4.46	4.53
$D_{QP\approx 35}$	2.73	2.93

Table 5.3 – *Comparaison à débit équivalent des versions classiquement décodées et synthétisées. Notes moyennes pour la séquence* Container

à une unité de l'échelle des dégradations. Ces tests permettent donc de valider l'hypothèse d'une qualité visuellement similaire, avec une tolérance de 3.7% en moyenne et un écart maximum de 6.66%, obtenu à bas débits sur la séquence *Container*.

Le tableau 5.3 permet de comparer les versions synthétisées et entièrement décodées par H.264 en utilisant des QP qui permettent d'approximer des débits équivalents. Aux trois débits proposés les notes données permettent de montrer que sur cette séquence, l'approche permet de gagner en qualité, à débit équivalent, mais surtout que le risque de perdre en qualité à débit égal est écarté.

Le tableau 5.4 permet de quantifier l'apport évident de la synthèse spatio-temporelle présentée dans ce chapitre, par rapport à une synthèse dite *intra*, effectuée indépendamment image par image. On constate effectivement que les écarts sont énormes, principalement dus à l'effet de scintillement, prédominant sur les zones synthétisées.

La tolérance moyenne de 3.7%, vérifiée par ces tests subjectifs, valide donc l'hyspothèse d'une compression opérée à qualité visuelle similaire. L'approche est alors capable de préserver plus de 30% de débit, par rapport au codage H.264 seul, sur certaines séquences et sous cette hypothèse. Il faut néanmoins être vigilant sur les paramètres de segmentation, directement responsables de la qualité des vidéos finales. En effet, si la définition des régions est trop tolérante, les algorithmes de synthèse ne seront pas à même de reconstruire

	synthèse *intra*	synthèse *inter*
QP=25	2.0	4.67
QP=35	1.73	3.47

Table 5.4 – *Comparaison entre la synthèse spatio-temporelle* (inter) *et la synthèse* (intra)*. Notes moyennes pour la séquence* Coastguard.

des zones « pas assez stationnaire ».

5.7 Conclusion.

Ce chapitre détaille une solution temporelle de compression d'un groupe d'images en utilisant la synthèse de texture temporelle, étroitement liée à un outil de compensation de mouvement. Deux segmentations sont maintenant appliquées. La segmentation temporelle sur le GOP permet d'abord de définir les régions cohérentes au sens du mouvement. L'approche de compensation en mouvement utilise une méthode affine, permettant de modéliser les mouvements globaux des régions segmentées. Dans l'implémentation actuelle, le nombre de régions est déterminé par l'utilisateur. La segmentation spatiale, présentée dans le chapitre 3, et dont l'utilisation est décrite dans le chapitre 4, permet toujours de déterminer les régions texturées qu'il est possible de synthétiser. L'implémentation actuelle de la méthode de segmentation se faisant image par image, elle est appliquée sur l'image centrale de chaque GOPM. Ainsi, si des régions texturées sont présentes au long du GOP, et rejetées par la segmentation temporelle pour être interpolées, elles deviennent candidates à la synthèse de texture.

Ainsi, les régions obtenues peuvent soit :
- être interpolées puisque cohérentes et rigides au long du GOP,
- être synthétisées, puisque classifiées comme textures qu'il est possible de synthétiser mais pas d'interpoler,
- être classiquement encodées dans les cas restants.

La synthèse de texture inspirée des méthodes pixel et patch présentées dans le chapitre précédent permet alors de traiter le second type de régions. Des adaptations temporelles limitent, via l'utilisation de la compensation en mouvement et de l'utilisation des images de référence, les artefacts de scintillement entre les images.

La campagne de tests subjectifs montre que la synthèse spatio-temporelle ne dégrade pas les vidéos de manière sensible avec une tolérance acceptable sur un panel de 17 observateurs. Cette méthode d'évaluation constitue la seule métrique viable à l'heure actuelle, des travaux restent à produire dans l'étude objective et automatique de la qualité des textures construites.

Conclusion et perspectives

Rappel des travaux réalisés

Le sujet de départ de ces travaux de thèse prévoyait l'application de techniques de synthèse d'images à la compression vidéo. L'objectif principal étant d'avoir un schéma complet fonctionnel, une première étape a consisté à définir un schéma global permettant un tel couplage. Ce dernier fait la part entre les textures qui pourront être reconstruites par synthèse au décodeur et le reste des images, qui est codé et transmis de manière classique. Cette reconstruction prend deux formes générales : la synthèse de texture et l'interpolation en mouvement. Dans les deux cas, les outils utilisés fonctionnent par région et non plus par macrobloc, les paramètres nécessairement transmis concernent ainsi des zones plus larges que des blocs 16×16. Ils sont ainsi très peu coûteux par rapport au débit total.

De tels schémas existent dans la littérature, dont les plus aboutis [NNHW07, ZSWL08], montrent clairement les promesses de ce type d'approche. Ces travaux de thèse tentent alors de proposer un schéma différent en ce qui concerne les outils utilisés, et notamment par l'observation de la complémentarité des algorithmes orientés pixel et patch suivant le type de texture traité.

Plusieurs outils ont été étudiés dans ce sens et implantés afin de remplir les différentes étapes du schéma, à savoir :

- des outils d'analyse : segmentation et caractérisation des textures,
- des outils de synthèse adaptés aux contraintes des régions segmentées,
- la possibilité de passer d'une méthode de synthèse à l'autre suivant le type de texture,
- des outils d'adaptation du schéma au domaine temporel à base d'estimation affine du mouvement des régions.

La construction du schéma temporel est ainsi issue d'un cheminement triennal incluant d'abord la tentative d'un schéma de raffinement puis l'élaboration d'une méthode dédiée aux images fixes. Couplée actuellement au standard H.264, son efficacité en termes de gains en débit provient de la capacité du schéma à forcer le mode skip pour un maximum de macroblocs, tout en minimisant les artefacts produits.

Les résultats visuels nous permettent actuellement de paramétrer une segmentation aboutissant à des gains voisins de 30% sur des séquences CIF et SD, pour les hauts débits. Les tests subjectifs permettent de valider l'hypothèse faite pour déterminer les gains présentés, à savoir une qualité des séquences reconstruites, visuellement similaire à celle des vidéos classiquement codées et reconstruite par un codec H.264.

Tous ces outils ont été retenus ou construits avec le souci principal de proposer un schéma cohérent au niveau de l'analyse et de la synthèse des textures. Malgré cela, la robustesse de ce type d'approche constitue toujours le principal obstacle à leur développement et utilisation, couplé aux difficultés d'évaluer les vidéos reconstruites. La section suivante permet de lister non exhaustivement les pistes en cours ou futures à creuser afin de valoriser et légitimer ce type de schéma dans le domaine de la compression de vidéo.

Travaux restants et perspectives

Une multitude de pistes permettrait d'améliorer l'efficacité et la robustesse des schémas intra et inter proposés dans ce document. De fait, chaque *brique* contenue dans le schéma global peut être optimisée voire totalement remise en cause. Parmi ces pistes, on pense notamment aux idées suivantes.

- La première continuité évidente serait d'implémenter de tels outils en lien avec un codeur HEVC, afin de profiter de l'efficacité de ce nouveau standard en construction, mais aussi de pouvoir comparer ce type d'approche avec l'état de l'art actuel en compression vidéo.
- De nombreux travaux sur l'attention visuelle sont déjà utilisés pour détecter les régions d'intérêt dans les vidéos. Il est ainsi possible d'allouer plus de débit pour ces objets ou zones de l'image puisque l'attention de l'observateur se concentre principalement sur ces régions saillantes. L'utilisation de telles *cartes de saillance* pourrait ainsi permettre de privilégier la synthèse des régions les moins regardées.
- Être capable de traiter des régions de taille et forme quelconques, en intra comme en inter serait aussi un atout. L'ordre de synthèse adopté permet déjà de synthétiser des régions aux contours atypiques. Cependant, cette propriété requiert la localisation de patchs pertinents, toujours efficaces pour contenir les informations adéquates.
- Une énergie non négligeable de ces travaux de thèse a été dédiée à l'implémentation et surtout l'adaptation en vain des algorithmes de synthèse de type EM au contexte du schéma de compression. La qualité des synthèses produites par ce type d'approche reste cependant prometteuse, malgré les contraintes évidentes liées à l'aspect itératif de ces algorithmes. If faudrait alors un schéma basé contenu complet et affranchi du standard.
- La prise en compte du mouvement dans le domaine temporel semble aussi perfectible. Si le modèle affine nous apparaît comme un bon compromis pour décrire le mouvement de régions rigides, le mode de segmentation et de détection itérative des mouvements principaux peut être amélioré. L'automatisation de la segmentation d'un nombre pertinent de régions paraît notamment importante à développer.
- Les techniques de filtrage temporel peuvent prévenir les artefacts de scintillement ou « flickering ».
- Sur les aspects synthèse de texture, des optimisations couplées aux compromis entre efficacité et complexité doivent être menées. Les travaux de cette thèse à vocation prospective sont focalisés sur les algorithmes mis en œuvre, en amont des contraintes de complexité.
- Il est possible d'améliorer la règle d'évaluation la confiance des pixels à synthétiser. Lors de la synthèse, chaque pixel ou patch est retenu grâce à la minimisation d'un écart. Ce dernier pourraît alors être pris en compte afin de propager les zones d'écart minimum, sensées contenir des motifs pertinents pour la suite de la synthèse. Aussi, les cartes utilisées dans le traitement de séquences pourraient par exemple tenir compte de la confiance des pixels colocalisés dans les images voisines. Cette idée

constitue une piste qui n'apparaissait pas comme prioritaire étant donné l'efficacité actuelle de la carte adoptée, mais pourrait permettre de minimiser les artefacts dus à l'ordre de synthèse.

Les schémas de compression orientés contenu tels que ceux présentés dans ce document sont très prometteurs, malgré leur manque de maturité. En effet, les schémas hybrides actuellement utilisés et qui continuent d'être développés avec HEVC commencent à atteindre leurs limites. Ces méthodes à base de synthèse sont néanmoins confrontées au domaine de l'évaluation de la qualité des images et des vidéos, qui est toujours fondée sur des métriques objectives, comparant les pixels décodés aux pixels originaux. Ces méthodes ne laissent aucune chance aux algorithmes de synthèse qui produisent des surfaces visuellement similaires, mais totalement différentes au niveau de la comparaison pixel à pixels. Ainsi, tant qu'une méthode automatique de détection de tels artéfacts n'aura pas été validée par un groupe d'experts, ce type de schéma ne pourra pas s'imposer, ni même concourir. La première étape passera par la remise en cause du PSNR dont l'immense majorité des acteurs de la compression s'accorde à décrier l'efficacité. Une fois cette barrière levée, ce type d'approche mais aussi tous les outils non fondés sur des critères débit/distorsion au sens du PSNR pourront compléter les méthodes existantes dans les standards H.264 et HEVC.

Enfin, l'idée principale d'utiliser plusieurs algorithmes de synthèse et de les mettre en compétition rappelle la sélection des modes de prédiction opérée dans les principaux standards de compression. Cette sélection a fait ses preuves avec l'optimisation débit/distorsion, et il apparaît légitime de penser aussi en termes de modes en compétition, suivant le type de région, pour ce type d'approche. D'autres travaux, par exemple, portent sur l'utilisation d'algorithmes paramétriques, très efficaces pour certains types de texture. Plus la gamme des textures correctement synthétisées sera large, plus nombreux seront les macroblocs forcés au skip par l'analyse/segmentation opérée au codeur.

ANNEXE A

Schéma de raffinement de texture.

Cette section vise à décrire un schéma global de compression axé sur l'utilisation d'une méthode de raffinement de texture. Il s'agit donc de répartir au mieux la qualité transmise des régions de l'image : les régions les plus pertinentes seront encodées avec une qualité supérieure parce qu'elles pourront ensuite servir à raffiner les autres. Dans la suite, on appellera *Haute Qualité* (HQ) la qualité de ces régions et *Basse Qualité* (BQ) celle des autres. Idéalement, le nombre (et la taille) des régions HQ sera faible, et d'un autre côté, on peut imaginer ne rien transmettre pour certaines régions BQ qui seraient entièrement reconstruites à l'aide des HQ. Pour ce faire, les régions HQ devront être représentatives des textures de l'image.

A.1 Synopsis.

Cette approche est constituée de trois étapes distinctes, développées ci-après :

- La constitution d'un *dictionnaire* de patchs représentatifs.
- L'encodage de la séquence avec un codeur classique.
- Le décodage et le *raffinement* des textures.

Cette structure, illustrée en figure A.1, représente une chaîne entière de codage-décodage. Elle peut cependant représenter uniquement la chaîne d'encodage. En effet, le fait de tester la sortie et d'en mesurer la qualité peut être réalisé au codeu afin d'améliorer les choix

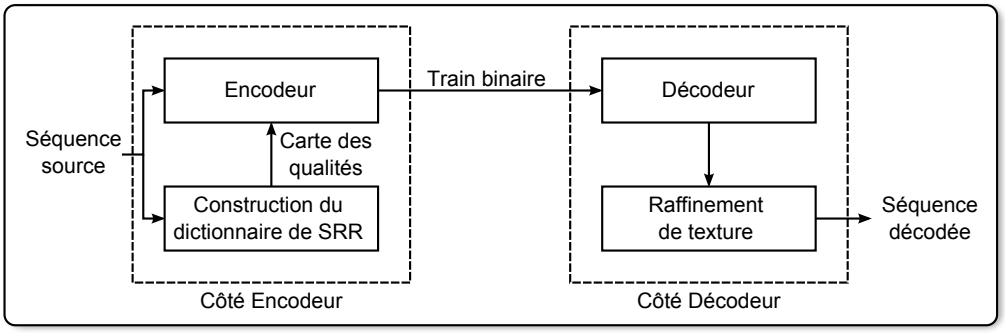


Figure A.1 – *Synopsis d'un schéma d'encodage avec raffinement de texture*

effectués, c'est pourquoi le schéma est bouclé sur la figure A.1.

Les régions pertinentes permettant de raffiner le maximum du reste des régions texturées sont dorénavant appelées SRR pour Sous-Régions Représentatives. Le paragraphe suivant permet de poser la problématique au codeur, c'est à dire les équations permettant de déterminer les SRR.

A.2 Fonctionnelle à maximiser.

L'évaluation de la pertinence d'une région texturée est réalisée en mesurant l'information qu'elle est capable de transmettre aux autres qui lui ressemblent, et qui ont été encodées avec une qualité plus faible afin de préserver du débit. Pour formaliser le problème, définissons la fonctionnelle à maximiser qui correspond à la somme des énergies qu'un ensemble de blocs est capable de fournir au reste de l'image. Ces énergies correspondent aux gains en qualité ΔQ des régions raffinées en texture selon une méthode de raffinement et la région HQ déterminée. Soient Π l'ensemble des régions de l'image à traiter et Ω un ensemble de SRR potentielles. Cette fonctionnelle s'écrit

$$\max_{\Pi} \left(\sum_{i \in \Omega} \Delta Q(r(i)) + \sum_{j \notin \Omega} \Delta Q(r(j)/\Omega) \right) \tag{A.1}$$

où Q est un critère de qualité et $r(i)$ la région texturée à la position i. Elle contient donc deux termes :

- le premier correspond à l'augmentation de la qualité des SRR elles-mêmes puisqu'elles seront encodées en HQ si elles sont choisies,
- le deuxième correspond à l'évolution de la qualité des autres régions après raffinement.

De cette étape est produite une carte de qualité localisant les régions représentative, qui servira au codeur comme paramètre d'entrée.

Soit $D[r_1, r_2]$ une métrique de distorsion entre deux régions r_1 et r_2. On introduit aussi $m(.)$ l'opérateur de raffinement. Les deux termes développés dans l'équation A.1 deviennent alors

$$\max_{\Pi} \big(f(i) + g(j) \big) \tag{A.2}$$

avec

$$\begin{cases} f(i) = \sum\limits_{i \in \Omega} \Big(D\big[r_{Q_0}(i), r_{ref}(i)\big] - D\big[r_{Q_1}(i), r_{ref}(i)\big] \Big) \\ g(j) = \sum\limits_{j \notin \Omega} \Big(D\big[m(r_{Q_1}(j), r_{Q_0}(i_{best}(j))), r_{ref}(j)\big] - D\big[r_{Q_1}(j), r_{ref}(j)\big] \Big) \end{cases} \tag{A.3}$$

où $f(i)$ représente la différence de qualité suivant $D[,]$ aux positions des SRR, et $g(j)$ le gain en qualité résultant au niveau des textures raffinées. $i_{best}(j)$ représente la position du meilleur correspondant pour la région à la position j, Q_1 et Q_0 sont les critères de qualités pour les SRR et les autres régions respectivement.

La section suivante décrit la manière de raffiner les parties dégradées.

A.3 Méthode de raffinement.

Résoudre l'équation nécessite de connaître la manière d'utiliser les SRR afin de rehausser la qualité des régions dégradées. Plusieurs techniques sont capables de résoudre

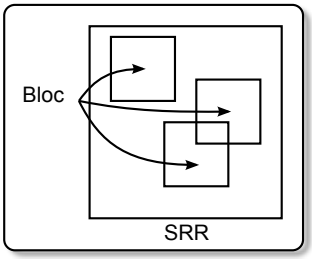


Figure A.2 – *Blocs à l'intérieur d'une région représentative*

ce problème. Une méthode fondée sur l'ajout de hautes fréquences est proposée dans ce chapitre : les algorithmes de synthèse décrits dans ce document peuvent aussi être mis en place après adaptation au contexte. La méthode proposée est fondée sur le fait que la quantification la plus sévère s'attaque particulièrement aux hautes fréquences dans les schémas d'encodage classiques tels que H.264 ou JPEG. En effet, lors de l'encodage des régions dégradées, la quantification est opérée dans le domaine DCT, privilégiant la précision des basses fréquences, fondamentales pour la perception humaine.

Ce raffinement ne va donc pas directement utiliser les SRR pour l'application sur l'image de sortie : un sous-bloc inclus dans la SRR choisie est d'abord sélectionné. Ainsi, pour chaque bloc à raffiner, chaque sous-bloc inclu dans la SRR va être considéré pour trouver la position maximisant la correspondance avec le bloc en cours. Le raffinement s'opère donc comme suit :

1. Calcul des coefficients de la DCT-2D du bloc en cours de raffinement, de taille $a \times a$ avec $a < r_{size}$.
2. Pour chaque bloc $a \times a$ contenu dans la SRR :
   - calcul des coefficients DCT,
   - application du raffinement donné par

$$\begin{cases} \forall n \in [0, \beta[, & \Lambda_{fus}(n) = \Lambda_{QP_0}(n) \\ \forall n \in [\beta, a^2 - 1], & \Lambda_{fus}(n) = \Lambda_{QP_0}(n) + \alpha\big(\Lambda_{QP_1}(n) - \Lambda_{QP_0}(n)\big) \end{cases} \tag{A.4}$$

   avec $\Lambda(n)$ correspondant au coefficient DCT à la position n dans l'ordre zigzag.

Deux degrés de liberté sont introduits ici :

- $\alpha \in]0; 1]$ est le poids appliqué à l'information ajoutée.
- $\beta \in [0; a^2 - 1]$ représente la position du premier coefficient DCT à fusionner dans l'ordre zigzag.

La recherche des SRR proposée, décrite dans la partie suivante, est une méthode a posteriori qui teste l'énergie apportée par une région à une autre, maximisant la fonctionnelle choisie.

A.4 Sélection des régions représentatives.

Le raffinement des texture dégradée nécessite donc de localiser précisément les blocs pertinents à l'intérieur des SRR. La figure A.2 montre un exemple de SRR carrée de taille $l \times l$, avec 3 blocs possibles de raffinement de taille $a \times a$. Une SRR contient donc $N = (l - a)^2$ blocs à la résolution des pixels entiers.

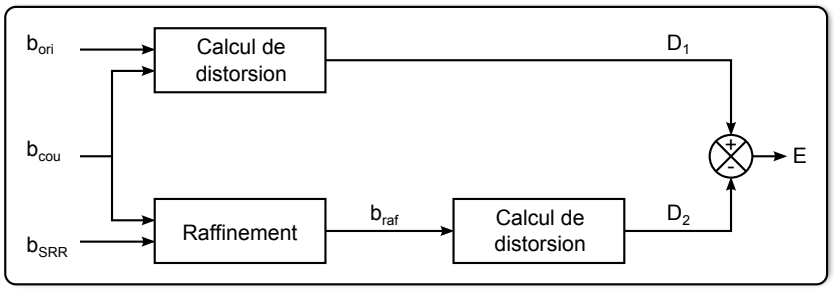


Figure A.3 – *Calcul de l'énergie pour la recherche des blocs représentatifs*

Deux approches ont été testées afin de sélectionner ces blocs de raffinement. Une première méthode exhaustive, destinée à évaluer le potentiel de l'application, et une méthode fondée sur la corrélation croisée.

A.4.1 Recherche exhaustive.

Cette section vise à rechercher le gain potentiel de cet algorithme en pratiquant une recherche exhaustive des blocs du dictionnaire, c'est à dire sur une grille de pixels et non une grille de blocs 4×4, 8×8 ou 16×16. Le gain est qualifié de potentiel puisque les décisions sont faites à l'aide de calculs de distorsions par rapport à l'image source. La recherche exhaustive permet donc d'assurer la localisation qui maximise l'adéquation entre le bloc à raffiner et son meilleur correspondant dans le dictionnaire. La figure A.3 montre le processus de calcul de l'énergie qui va permettre de choisir les blocs et donc les SRR à encoder avec la qualité HQ.

Trois blocs sont pris en compte dans ce calcul : $b_{SRR} \subset SRR$ de qualité HQ, le bloc courant b_{cou} à la qualité BQ, ainsi que le bloc b_{ori} colocalisé à b_{cou} dans l'image originale. L'opération de fusion s'applique donc potentiellement entre b_{SRR} et b_{cou}. Le schéma calcule alors la SSE comme métrique de distorsion D. Deux distorsions D_1 et D_2 sont calculées avec le bloc d'origine comme référence : D_1 pour le bloc courant encodé à BQ b_{cou}, et D_2 pour le nouveau bloc b_{raf} issu de la fusion entre b_{SRR} et b_{cou}. La différence $E = D_2 - D_1$ permet alors d'évaluer l'énergie apportée par b_{SRR} à b_{cou}. Une fois que tous les blocs $b_{SRR} \subset SRR$ ont été testés, on garde

$$b_{best} \text{ tel que } E(b_{best}) = \max_{b_{SRR} \subset SRR} (E(b_{SRR})) \tag{A.5}$$

puis l'énergie potentiellement apportée par une SRR au reste de l'image I est donnée par

$$E_{SRR} = \sum_{b \in I\{\}SRR} E\left(b_{best}(b)\right). \tag{A.6}$$

A.4.2 Corrélation croisée.

Une première solution alternative à la recherche exhaustive, qui n'est pas crédible en termes de coûts de calculs, est proposée ici. Cette solution permet de plus une recherche de corrélation sans l'utilisation de l'image source, et peut par conséquent être utilisée au codeur comme au décodeur. Afin de rechercher la position exacte de $b_{SRR} \subset SRR$, maximisant la corrélation avec b_{cou} la corrélation croisée suivante est calculée :

$$C(i,j) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} b_I(k,l) b_{SRR}(i+k, j+l) \tag{A.7}$$

où C est un tableau à deux dimensions, b_I et b_SRR sont des blocs appartenant à l'image courante et au dictionnaire respectivement. Les coordonnées i_p et j_p telles que $C(i_p, j_p)$ est maximum correspondent au *pic de corrélation* recherché. Le calcul de la corrélation croisée se fait dans le domaine de Fourier via :

$$C = TFD^{-1}(TFD(b_I).TFD(b_D)^*), \tag{A.8}$$

où TFD^* correspond au conjugué de la transformée de Fourier discrète. On a alors

$$TFD(b_D)^* = TFD(b_D) \tag{A.9}$$

puisque b_D est un signal à valeurs réelles. La valeur correspondante de $C(i_p, j_p)$ ainsi que le vecteur permettant de localiser (i_p, j_p) sont ensuite utilisés pour déterminer l'opération de fusion.

Afin d'éviter les repliements de spectre, une fenêtre de Blackman est appliquée aux deux blocs comparés par

$$b_f(i,j) = Bl(i,j).b(i,j) \tag{A.10}$$

où b_f est le bloc fenêtré résultant et la fenêtre de Blackman Bl est donnée par

$$\begin{cases} Bl(i,j) = B(i).B(j) \\ B(n) = 0.42 - 0,5.cos(2\pi n) + 0,08.cos(4\pi n)), \end{cases} \tag{A.11}$$

Du côté du décodeur, les pics de corrélation peuvent être comparés au maximum de la fonction d'autocorrélation du bloc courant $A_{b_{cou}}(0,0)$. Un seuillage par rapport à cette valeur peut ensuite permettre d'évaluer le niveau de corrélation entre le bloc du dictionnaire et le bloc courant. Il faut alors définir un coefficient α permettant de définir le seuil

$$\lambda = \alpha A_{b_{cou}}(0,0). \tag{A.12}$$

Ainsi α représente un pourcentage de confiance au dessus duquel on décide d'appliquer la fusion permettant le raffinement du bloc.

A.5 Tri des atomes du dictionnaire.

Après avoir calculé l'énergie fournie par chaque SRR potentielle, il est nécessaire de les trier afin de construire le dictionnaire des régions qui seront encodées avec la qualité HQ. Le dictionnaire est construit itérativement, SRR par SRR. Après avoir ajouté la région la plus pertinente trouvée, toutes les parties de l'image pour lesquelles cette SRR est la plus corrélée sont retirées du calcul d'énergie. Après la détection du premier atome constituant le dictionnaire Ω_1, le prochain « meilleur » atome est sélectionné à partir de

$$\forall j \notin \Omega_1 / i_{best}(j) \in \Omega_1, D\big[m(r_{HQ}(j), r_{BQ}(i_{best}(j))), r_{ref}(j)\big] = 0, \tag{A.13}$$

avec $Card(\Omega_1) = 1$. Ce processus itératif peut alors être interrompu par un seuil de distorsion ou un nombre d'atomes pour le dictionnaire fixé en fonction des dimensions de l'image.

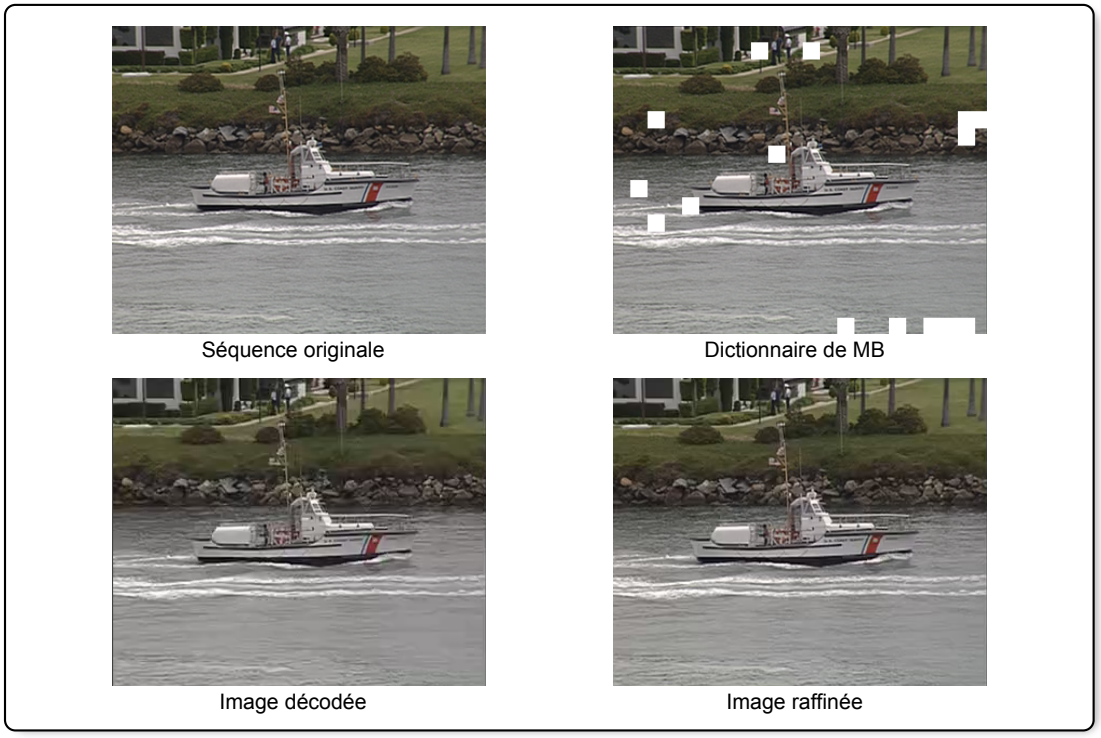


Figure A.4 – *Images correspondant aux différentes étapes du schéma.*

A.6 Encodage.

L'encodage suit la carte des qualités envoyée par l'étape de construction du dictionnaire de SRR. Les SRR sont encodées avec la qualité HQ et le reste de l'image à la qualité BQ. Dans le cadre d'un codeur H.264, il est possible de déterminer le paramètre de quantification (QP) pour chaque MB. Si les SRR sont des MB, il est alors possible d'éviter d'envoyer des informations supplémentaires dans le train binaire, permettant de détecter la qualité de chacun des MB reçu. Dans d'autres cas de figure, une information de bord est codée et envoyée à cette fin.

A.7 Analyse expérimentale.

Le schéma décrit précédemment a été implémenté en combinaison d'un codeur type H.264. Les SRR correspondent dans la suite à des MB. Une option permettant de cartographier le paramètre QP, suivant les MB des images, a été intégrée dans le codeur JM [jvt07]. Bien que le schéma convienne au codage inter-images, les expérimentations exposées ont été menées sur un schéma *intra*. Ainsi les séquences encodées par le codeur H.264 seront composées d'images I uniquement. La construction du dictionnaire est appliquée en générant deux versions des images : une HQ et une BQ, en utilisant deux paramètres QP_1 et QP_0. Les SRR sont alors extraites selon les techniques proposées. Dans la suite, deux séquences de test CIF *Coastguard* et *Macleans* sont utilisées pour valider les schémas. La figure A.4 illustre les différentes étapes du schéma avec les MB sélectionnés en blanc, avec ensuite les images décodées avant et après raffinement.

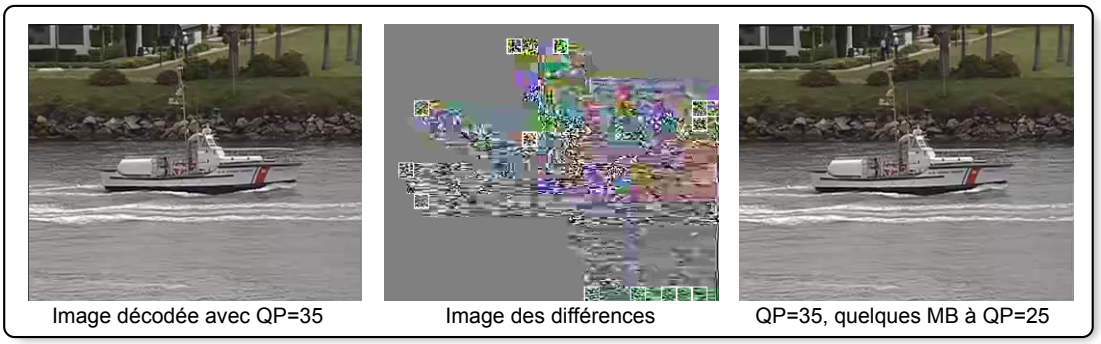


Figure A.5 – *Différences entre une image entièrement encodée avec* $QP = 35$ *et une image à* $QP = 35$ *avec quelques MB encodés avec* $QP = 25$.

Gains	blocs 4×4	blocs 8×8
ΔPSNR	2.46dB	0.85dB
Δdébit	-17.62%	-6.62%

Table A.1 – *Gains potentiels moyens en débit et distorsion en utilisant des blocs* 4×4 *et* 8×8.

A.7.1 Remarques préliminaires.

Le standard H.264 permet de faire varier le paramètre QP à l'intérieur d'une même image. Cependant, le fait d'encoder certains MB avec une qualité différente affecte les MB anti-causaux, c'est à dire ceux qui seront potentiellement prédits à l'aide des premiers. Cet effet est illustré par la figure A.5 où les différences ont été accentuées pour être facilement discernables. Dans le cas où il n'y aurait aucun raffinement de texture après cette étape, ce schéma donne des résultats en deçà de l'encodage avec un unique QP. Ceci est dû à l'ordre du décodage de l'image qui induit le passé causal de la prédiction. Le débit alloué à ces MB ne permet donc d'améliorer qu'une partie de l'image. Ainsi, pour la séquence CIF *Coastguard* encodée avec $QP_0 \in [20, 45]$ et $QP_1 = QP_0 - 10$, on observe une augmentation moyenne du débit de 6.54% en moyenne, en utilisant la méthode de calcul de [Bjo01]. Un risque est donc pris ici de diminuer temporairement l'efficacité de codage de H.264 pour gagner ensuite en qualité grâce à l'étape de raffinement.

A.7.2 Potentiel de l'approche.

La figure A.6 et les tableaux A.2 et A.3 présentent les résultats obtenus avec un nombre d'atomes fixé à $Card(\Omega) = 15$. Des tailles de blocs de 4×4 et 16×16 ont été testées car elles permettent une intégration pratique avec le standard H.264. Le tableau A.1 montre lui que les meilleurs résultats proviennent des expérimentations utilisant des blocs de taille 4×4 pour le schéma de raffinement. L'utilisation de blocs 4×4, alors que les atomes préservés sont des macroblocs de taille 16×16, permet une meilleure localisation du meilleur bloc correspondant. On remarque que cette taille de bloc permet de réduire le débit de 17.62% à PSNR égal, alors que l'utilisation de blocs 8×8 réduit de 6.62%.

Le tableau A.2 présente les gains potentiels de l'approche avec $\Delta QP = QP_0 - QP_1 = 10$. La troisième colonne présente notamment le gain moyen sur les 4 QP_0 les plus faibles avec par exemple un gain en débit de 42.17% sur la séquence *Coastguard* par rapport à la

Data	Gains	Bas débits	Hauts débits	Moyenne
Coastguard	ΔPSNR	2.71dB	2.83dB	2.77dB
	Δbit-rate	-42.17%	-30.45%	-36.31%
Macleans	ΔPSNR	2.25dB	1.80dB	2.02dB
	Δbit-rate	-34.45%	-20.92%	-27.68%

Table A.2 – *Gains potentiels moyens sur les 4 QP correspondant aux hauts débits et les 4 QP bas débits pour les séquences CIF* Coastguard *et Macleans.*

Gains	$\Delta QP = 5$	$\Delta QP = 10$	$\Delta QP = 15$
ΔPSNR	1.95dB	1.59dB	0.97dB
Δdébit	-27.43%	-23.18%	14.34%

Table A.3 – *Gains potentiels pour la séquence* Coastguard *suivant l'écart* ΔQP .

version encodée en intra par le standard H264/AVC.

Le tableau A.3 permet de montrer l'impact du choix du ΔQP sur les gains observés pour la séquence Coastguard, toujours comparés à une référence H.264 en mode intra. Les limites de cette approche sont mises en évidence par le fait que le minimum d'écart donne les meilleurs résultats. C'est en effet avec un $\Delta QP = 5$ que le gain monte à 27.43% de débit préservé.

A.7.3 Division en cadrans.

Afin de présenter une approche plus réaliste en termes de coûts de calculs, il a été décidé d'appliquer une segmentation grossière sur les images. Afin de tester rapidement une solution simple, cette dernière revient à découper les images en cadrans réguliers, afin de localiser spatialement la recherche des atomes du dictionnaire. La figure A.6 montre ainsi un exemple de découpage avec la sélection d'un atome par cadran. Les dictionnaires et images résultantes sont présentés pour la séquence *Macleans* avec $QP_0 = 40$ et $\Delta QP = 10$.

Dans le cas d'une recherche exhaustive, malgré le côté idéal de l'énergie que les atomes sont capables de transmettre, on remarque qu'il est possible qu'ils soient condensés spatialement. Il sera ainsi difficile, côté décodeur, d'appliquer la même recherche de l'atome correspondant le mieux. L'expérience des cadrans permet donc d'accéder directement à l'échantillon d'atomes contenu dans le cadran en cours, accélérant le processus et assurant le choix d'un atome spatialement cohérent. La figure A.7 présente le potentiel des gains pour la méthode des cadrans, comparée à la méthode exhaustive ainsi qu'à la référence H.264 en mode intra. On note que malgré les contraintes imposées, la localisation par région permet une simplification du processus tout en conservant des gains prometteurs. La figure A.8 présente elle les résultats obtenus selon la différence entre le QP des SRR et du reste de l'image. Les courbes de débit-distorsions pour les 3 $\Delta QP = \{5, 10, 15\}$ sont proposées.

Afin de comparer les méthodes exhaustives et par cadrans, la figure A.7 présente les courbes débit/PSNR des deux approches à celle de la référence H.264 utilisant une prédiction avec les modes Intra. L'approche par cadrans donne, comme attendu, des résultats en deçà de l'approche exhaustive. L'approche reste cependant une simplification

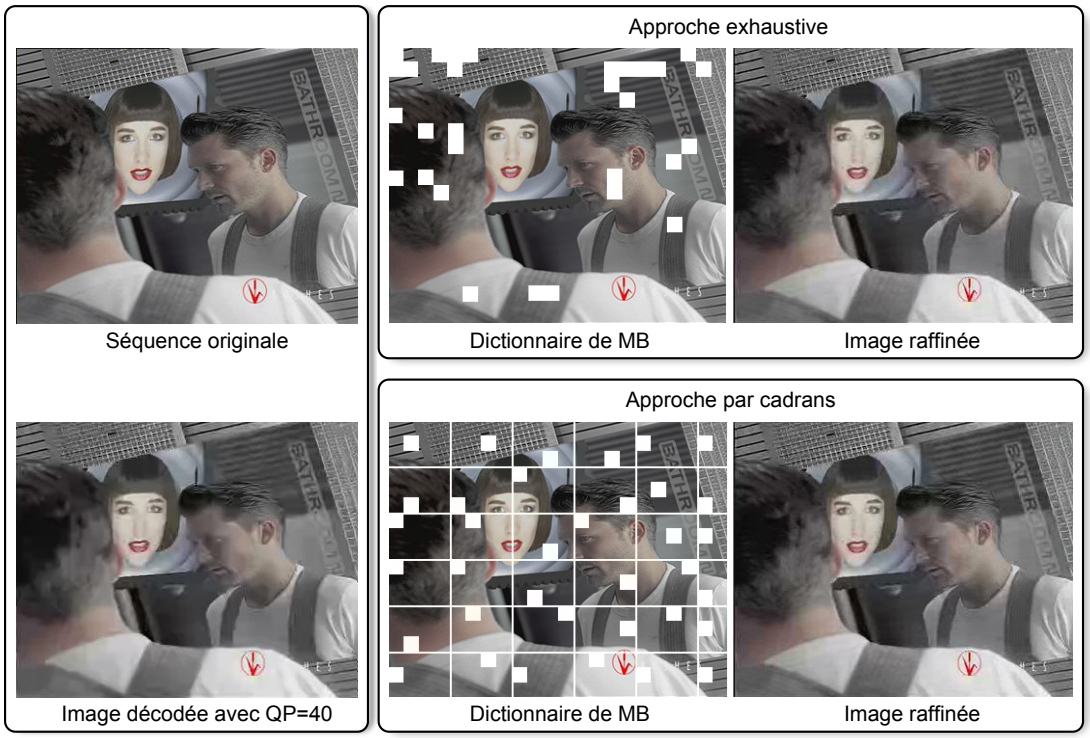


Figure A.6 – *Dictionnaires et images résultantes. La première ligne présente l'approche exhaustive. La seconde présente une approche utilisant des cadrans, séparés par les lignes blanches, dans lesquels le macrobloc le plus représentatif est préservé.*

prometteuse, en vue d'une réelle étape de segmentation. En effet, la baisse des performances n'est pas très importante, comparée à la réduction de coût engendrée, d'un facteur moyen de 35.8 sur les séquences CIF *Coastguard* et *Macleans*.

A.7.4 Corrélation croisée.

Cette section décrit l'utilisation de l'outil de corrélation croisée, précédemment introduit dans la section A.4. Cette étape permet de remplacer la recherche exhaustive au codeur et rend l'approche symétrique entre les processus côté codeur et décodeur

Le tableau A.4 permet d'évaluer les distorsions introduites par l'approche utilisant la corrélation croisée. Comme pour l'utilisation des cadrans, il est évident que cette approche sous-optimale induit des performances moindres en termes de débit/PNSR. Cependant, cet algorithme se base uniquement sur le signal encodé pour la comparaison et la mise en correspondance du bloc courant avec ceux du dictionnaire, alors que l'approche exhaustive requiert la version original pour le calcul de distorsion.

A.8 Discussion, limitations.

Ce schéma vise à exploiter des méthodes de raffinement de texture en vue de cartographier différemment le débit alloué à chaque région des images. L'idée première était de construire un schéma générique capable pour le codeur de créer cette cartographie indépendamment de l'outil de raffinement utilisé. Une première implémentation en combi-

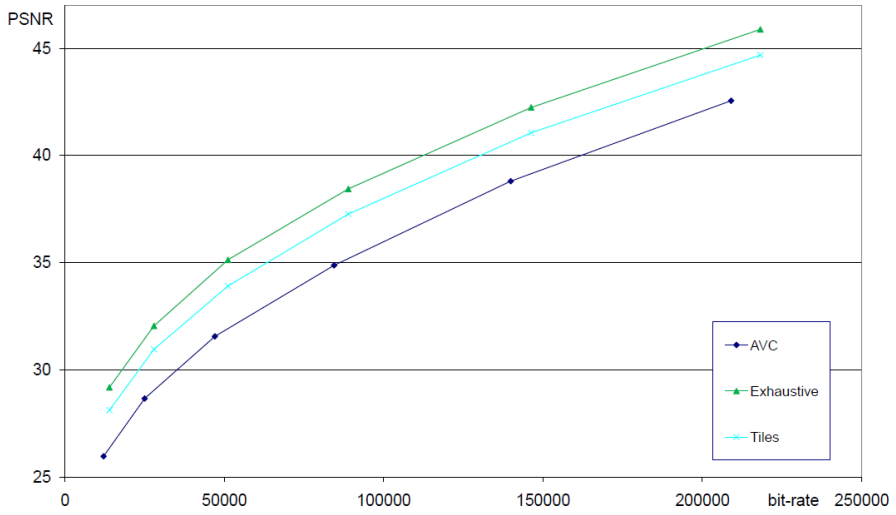


Figure A.7 – *Courbes débit distorsion pour la méthode exhaustive, la méthode par cadrans, et H.264/AVC en mode intra sur la séquence* Coasguard.

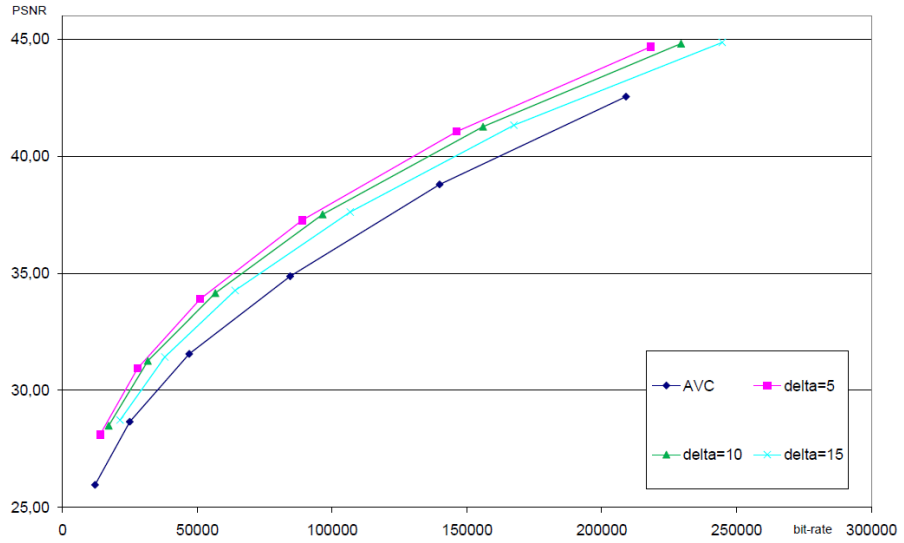


Figure A.8 – *Courbes débit/distorsion pour l'approche en cadrans comparée à H.264 suivant l'écart* ΔQP.

Gains	faibles débits	hauts débits	moyenne
ΔPSNR	0.37dB	0.31dB	0.34dB
Δdébit	-4.61%	-4.13%	-4.37%

Table A.4 – *Gains observés avec l'utilisation de la corrélation croisée sur la séquence* Coatsguard *avec* $QP_0 = 40$ *et* $\Delta QP = 10$.

naison avec un codeur H.264 a été développée. Fondée sur un ajout des coefficients DCT hautes fréquences, celle-ci n'a pas convaincu sur l'idée de coder quelques MB avec une meilleure qualité. En effet, afin de passer à la suite du cheminement souhaité, c'est à dire utiliser des outils de synthèse de texture, l'idée de rehausser la qualité de toute l'image, même des zones non-texturées parît délicate.

C'est donc l'inverse qui a été choisi pour la suite des travaux de thèse, présentés dans les chapitres 4 et 5, *i.e.* sacrifier uniquement les régions que l'on est capable de synthétiser et non encoder des patchs avec une qualité supérieure.

ANNEXE B

Image et séquences utilisées.

Cette annexe référence les images et séquences vidéo utilisées pour valider les différentes étapes des schémas décrits. Certaines sont des séquence couramment utilisée en traitement de vidéo, mais aussi d'images ou de séquences libres de droit ou personnelles. Lorsqu'il s'agit d'une séquence, une ou deux images représentatives sont présentées.

Raid Maroc 720 × 576

Soccer 704 × 576

Solidor 720 × 576

zebres 720 × 576

Images utilisées pour les tests subjectifs

Image 0

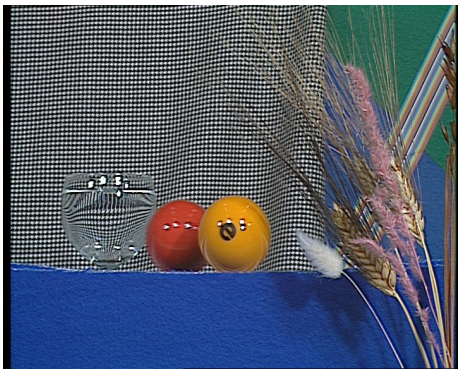
Image 133

Séquence Coastguard 352 × 288

Séquence Container 352 × 288

Basket 416 × 240

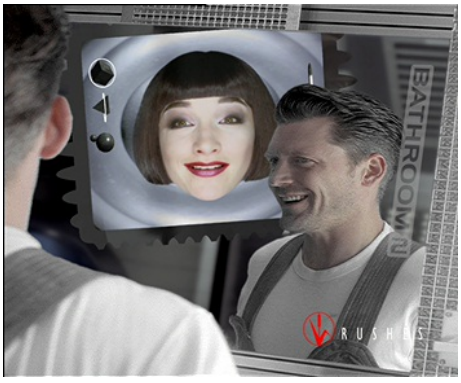
Dromadaire 1280 × 720



Snooker 720 × 576

Wool 720 × 576

Séquence Crowd 1280 × 720



macleans 352 × 288

titleist 352 × 288

vautours 720×576

ICE 704×576

patinage 352×288

Table des figures

Liste des tableaux

Publications personnelles

[1] F. Racapé, S. Lefort, E. François, M. Babel et O. Déforges. Adaptive pixel/patch-based synthesis for texture compression, *IEEE International Conference on Image Processing, ICIP*. 2011

[2] F. Racapé, S. Lefort, D. Thoreau, M. Babel et O. Déforges. Characterization and adaptive texture synthesis-based compression scheme, *European Signal Processing Conference, EUSIPCO*. 2011.

[3] F. Racapé, S. Lefort, J. Viéron, M. Babel et O. Déforges. Texture refinement framework for improved video coding, *SPIE Visual Information Processing and Communication*. 2010.

[4] F. Racapé, D. Thoreau, J. Viéron, A. Martin et G. Ombrouck, *Procédé de codage d'image avec synthèse de texture*. Brevet EP2281396, WO2009147224, 2011-02-09.

[5] F. Racapé, J. Viéron, M. Babel et O. Déforges. *Codage d'image avec raffinement de la texture en utilisant des sous-régions représentatives*. Brevet EP2268030 2010-12-29.

[6] D. Thoreau, E. François, J. Viéron et F. Racapé. *Procédé de décodage d'un flux représentatif d'une séquence d'images et procédé de codage d'une séquence d'images*. Brevet FR2948845, 2011-02-04.

[7] A. Martin, D. Thoreau, J. Viéron et F. Racapé. *Procédé de codage de données vidéo pour une séquence d'images basé sur un algorithme matching pursuit*. Brevet WO2010149554 2010-12-29.

[8] E. François, D. Thoreau, F. Racapé. *Procédé de décodage d'un flux représentatif d'une séquence d'images, procédé de codage d'une séquence d'images et structure de données codées*. Brevet WO2010086393 2010-08-05.

Bibliographie

[AA68] V.I. Arnol'd et A. Avez : *Ergodic problems of classical mechanics*, volume 48. Benjamin New York, 1968. 26

[Ade91] Edward H. Adelson : Layered representations for image coding. Rapport technique, 1991. 67

[Ash01] M. Ashikhmin : Synthesizing natural textures. *In Proceedings of ACM Symposium on Interactive 3D Graphics*, pages 217–226, 2001. 2, 19, 20, 21, 23, 24, 40, 41, 66, 112, 126, 142, 175, 176

[AVC03] (ITU-T Rec. H.264 — ISO/IEC 14496-10 AVC) : Draft-t recommandation and final draft international-standard of joint video specification, 2003. 66

[BD96] S.A. Basith et S.R. Done : Digital video, mpeg and associated artifacts. *Imperial College London*, 1996. 51

[Bes74] J Besag : Spatial Interaction and the Statistical Analysis of Lattice Systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):192–236, 1974. 11

[Bes86] J. Besag : On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):259–302, 1986. 14

[Bjo01] G. Bjontegaard : Calculation of average PSNR differences between RD-curves. *In ITU - Telecommunications Standardization Sector : VCEG-M33*, Austin, TX, 2001. 165

[Bro99] P. Brodatz : *Textures : a photographic album for artists and designers*. Dover Publications New York, 1999. 14

[Bru01] Eric Bruno : *De l'estimation locale à l'estimation globale de mouvement dans les séquences d'images*. Thèse de doctorat, Université Joseph Fourier, Grenoble, 2001. 132

[BSCB00] M. Bertalmio, G. Sapiro, V. Caselles et C. Ballester : Image inpainting. *In Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424, 2000. 37, 39, 71

[BVSO03] M. Bertalmio, L. Vese, G. Sapiro et S. Osher : Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing*, 12(8):882–889, 2003. 38, 39, 176

[BVZ99] Y. Boykov, O. Veksler et R. Zabih : Fast approximate energy minimization via graph cuts. *In The Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 377–384. IEEE, 1999. 28

[BZD07] M. Bosch, F. Zhu et E.J. Delp : Spatial texture models for video compression. *In IEEE International Conference on Image Processing, ICIP*, volume 1, pages I–93. IEEE, 2007. 67

[BZD08] M. Bosch, F. Zhu et E. Delp : Models for texture based video coding. *In Proceedings of LNLA, IEEE International Workshop on Local and Non-Local Approximation in Image Processing, Lausanne, Switzerland*, 2008. 67

[CCB85] R. Chellappa, S. Chatterjee et R. Bagdazian : Texture synthesis and compression using Gaussian-Markov random field models. *IEEE transactions on systems, man, and cybernetics*, 15(2):298–303, 1985. 12

[CD99] M.L. Comer et E.J. Delp : Segmentation of textured images using a multiresolution gaussian autoregressive model. *Image Processing, IEEE Transactions on*, 8(3):408–420, 1999. 82

[CJ83] G.R. Cross et A.K. Jain : Markov random field texture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):25–39, 1983. 11

[CK85] R. Chellappa et R. Kashyap : Texture synthesis using 2-D noncausal autoregressive models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33(1):194–203, 1985. 12

[Cor70] T.N. Cornsweet : Visual perception. 1970. 45

[CPT04] A. Criminisi, P. Pérez et K. Toyama : Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212, 2004. 39, 40, 114, 176

[CR68] FW Campbell et JG Robson : Application of Fourier analysis to the visibility of gratings. *The Journal of Physiology*, 197(3):551, 1968. 46, 176

[CS00] T. Chan et J. Shen : Mathematical models for local deterministic inpaintings. *UCLA CAM TR*, 2000. 38

[CS01] T.F. Chan et J. Shen : Nontexture inpainting by curvature-driven diffusions. *Journal of Visual Communication and Image Representation*, 12(4):436–449, 2001. 38

[Dal93] S. Daly : The visible differences predictor : an algorithm for the assessment of image fidelity, 1993. 64

[DB97] J.S. De Bonet : Multiresolution sampling procedure for analysis and synthesis of texture images. *In Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 361–368. ACM Press/Addison-Wesley Publishing Co., 1997. 14, 175

[DBBR07] O. Déforges, M. Babel, L. Bédat et J. Ronsin : Color lar codec : a color image representation and compression scheme based on local resolution adjustment and self-extracting region representation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(8):974–987, 2007. 102

[Déf04] Olivier Déforges : Codage d'images par la méthode lar et méthodologie adéquation algorithme architecture : de la définition des algorithmes de compression au prototypage rapide sur architectures parallèles hétérogènes. 2004. 97

[DH04] A. Dumitras et BG Haskell : An encoder-decoder texture replacement method with application to content-based movie coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(6):825–840, 2004. 68

[DHIC03] A. Dumitras, BG Haskell, A.C. Inc et CA Cupertino : A texture replacement method at the encoder for bit-rate reduction of compressed video. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(2):163–175, 2003. 68

[DL+58] DZN DE LANGE *et al.* : Research into the dynamic nature of the human fovea cortex systems with intermittent and modulated light. I. Attenuation characteristics with white and colored light. *JOSA*, 48(11):777–783, 1958. 46

[dLR77] A. P. dempster, N. M. Laird et D. B. Rubin : Maximum likelihood from incomplete data via the em algorithm. *journal of the royal Statistical Society*, 39(1):1–38, 1977. 31

[DV05] C. Demonceau et P. Vasseur : Champs de markov pour le traitement d'images catadioptriques. 2005. 11

[EF01] A. A Efros et W. T Freeman : Image quilting for texture synthesis and transfer. *In Proceedings of SIGGRAPH 2001*, pages 341–346, 2001. 27, 30, 40, 41, 71, 176

[EK98] T. Ebrahimi et M. Kunt : Visual data compression for multimedia applications. *Proceedings of the IEEE*, 86(6):1109–1125, 1998. 94

[EL99] Alexei Efros et Thomas Leung : Texture synthesis by non-parametric sampling. *In In International Conference on Computer Vision*, pages 1033–1038, 1999. 15, 16, 26, 33, 39, 114, 175

[FS05] MJ Fadili et J.L. Starck : Em algorithm for sparse representation-based image inpainting. *In IEEE International Conference on Image Processing, ICIP*, volume 2, pages II–61. IEEE, 2005. 38

[GC03] B. Georgescu et CM Christoudias : The edge detection and image segmentation (edison) system. *Robust Image Understanding Laboratory, Rutgers University.*, 2003. 71

[Geo06] T. Georgiev : Covariant derivatives and vision. *Computer Vision–ECCV 2006*, pages 56–69, 2006. 77

[GG86] S. Geman et C. Graffigne : Markov random field image models and their applications to computer vision. *In Proceedings of the International Congress of Mathematicians*, volume 1496, page 1517, 1986. 11

[Gir87] B. Girod : The efficiency of motion-compensating prediction for hybrid coding of video sequences. *Selected Areas in Communications, IEEE Journal on*, 5(7):1140–1154, 1987. 55

[GSX] B. Guo, H. Shum et Y.Q. Xu : Chaos mosaic : Fast and memory efficient texture synthesis. *Microsoft research paper MSR-TR-2000-32*. 25

[Har58] HO Hartley : Maximum likelihood estimation from incomplete data. *Biometrics*, 14(2):174–194, 1958. 31

[HB95] D.J. Heeger et J.R. Bergen : Pyramid-based texture analysis/synthesis. *In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM, 1995. 14

[HEV10a] Joint Call for Proposal on Video Compression Technology, 2010. 59

[HEV10b] Vision, Applications and Requirements for High-Performance Video Coding (HVC), 2010. 59

[HEV11] HM : reference Software for HEVC version HM-3.0, 2011. 59

[HS85] R.M. Haralick et L.G. Shapiro : Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985. 94

[HSD73] R.M. Haralick, K. Shanmugam et I. Dinstein : Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, 3(6):610–621, 1973. 82

[HTW07] H. D Huang, X. Tong et W. C Wang : Accelerated parallel texture optimization. *Journal of Computer Science and Technology*, 22(5):761–769, 2007. 2, 34, 112, 126

[IR93] R.B.T. ITU-R : 500, methodology for the subjective assessment of the quality of television pictures, 1993. 60

[ITU02] ITU : Recommendation ITU-R BT. 500-11. Methodology for the Subjective Assessment of the Quality of Television Pictures. *Standardization Sector of ITU*, 2002. 122

[JJ81] J. Jain et A. Jain : Displacement measurement and its application in interframe image coding. *Communications, IEEE Transactions on*, 29(12):1799–1808, 1981. 55

[Jul62] B. Julesz : Visual pattern discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, 1962. 7

[Jul81] B. Julesz : Textons, the elements of texture perception, and their interactions. *Nature*, 1981. 12

[jvt07] JM reference software version 14.0 (current version : 15.1), 2007. 121, 164

[KAK05] V. Kwatra, I. Essa Aaron et B. N. Kwatra : Texture Optimization for Example-based Synthesis. *In Proceedings of ACM SIGGRAPH*, pages 795 – 802, 2005. 2, 31, 32, 34, 35, 40, 75, 112, 126, 176

[KAK+07] V. Kwatra, D. Adalsteinsson, T. Kim, N. Kwatra, M. Carlson et M. Lin : Texturing fluids. *IEEE transactions on visualization and computer graphics*, pages 939–952, 2007. 34

[KBBN05] S. Kumar, M. Biswas, S.J. Belongie et T.Q. Nguyen : Spatio-temporal texture synthesis and image inpainting for video applications. *In IEEE International Conference on Image Processing, ICIP*, volume 2, pages II–85. IEEE, 2005. 40

[KDM02] S.L. Kilthau, M.S. Drew et T. Moller : Full search content independent block matching based on the fast fourier transform. *In International Conference on Image Processing.*, volume 1, pages I–669. IEEE, 2002. 40

[KGKH93] M. Kunt, G. Granlund, M. Kocher et C. Horne : *Traitement de l'information :(Volume 2, Traitement numérique des images)*. 1993. 44

[KGL+08] Aditya Khandelia, Saurabh Gorecha, Brejesh Lall, Santanu Chaudhury et Mona Mathur : Parametric video compression scheme using ar based texture synthesis. *In ICVGIP '08 : Proceedings of the 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 219–225, Washington, DC, USA, 2008. IEEE Computer Society. 68, 69, 177

[KSE+03] V. Kwatra, A.A. Schödl, I. Essa, G. Turk et A. Bobick : Graphcut textures : image and video texture synthesis using graph cuts. *In Proceedings of ACM SIGGRAPH*, pages 277–286, 2003. 2, 27, 28, 29, 30, 35, 40, 41, 71, 72, 74, 77, 112, 113, 126, 176

[KSS80] R. Kindermann, J.L. Snell et American Mathematical Society : *Markov random fields and their applications*. American Mathematical Society, 1980. 11

[Law80] K.I. Laws : Textured Image Segmentation. Rapport technique, University of Southern California Los Angeles Image Processing Inst., 1980. 8

[LGW03] Y. Lu, W. Gao et F. Wu : Efficient background video coding with static sprite generation and arbitrary-shape spatial prediction techniques. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(5):394–405, 2003. 76

[LH05] S. Lefebvre et H. Hoppe : Parallel controllable texture synthesis. *ACM Transactions on Graphics (TOG)*, 24(3):786, 2005. 23, 25, 176

[LHW+06] W.C. Lin, J. Hays, C. Wu, Y. Liu et V. Kwatra : Quantitative evaluation of near regular texture synthesis algorithms. 2006. 30

[Li95] S.Z. Li : *Markov random field modeling in computer vision*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1995. 11

[Lin05] Wen-Chieh Lin : *A Lattice-Based MRF Model For Dynamic Near-Regular Texture Tracking and Manipulation*. Thèse de doctorat, Robotics Institute, Carnegie Mellon University, 2005. 8, 9, 29, 175

[LLAY06] C.B. Liu, R.S. Lin, N. Ahuja et MH Yang : Dynamic textures synthesis as nonlinear manifold learning and traversing. *In Proc. British Machine Vision Conf*, volume 2, pages 859–868. Citeseer, 2006. 12

[LLH04] Y. Liu, W.-C. Lin et J. Hays : Near-regular texture analysis and manipulation. *In SIGGRAPH '04 : ACM SIGGRAPH 2004 Papers*, pages 368–376, New York, NY, USA, 2004. ACM. 9, 40, 84

[LLX+01] L. Liang, C. Liu, Y. Q Xu, B. Guo et H. Y Shum : Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (TOG)*, 20(3):150, 2001. 26, 28, 176

[LSW+07] Dong Liu, Xiaoyan Sun, Feng Wu, Shipeng Li et Ya-Qin Zhang : Image compression with edge-based inpainting. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(10):1273 –1287, oct. 2007. 71, 72, 76, 177

[LTL02] Yanxi Liu, Yanghai Tsin et Wen-Chieh Lin : The promise and the perils of near-regular texture. *International Journal of Computer Vision*, 62:1–2, 2002. 9, 29, 30

[LWBK02] L. Lu, Z. Wang, A.C. Bovik et J. Kouloheris : Full-reference video quality assessment considering structural distortion and no-reference quality evaluation of MPEG video. *In Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 61–64. IEEE, 2002. 62

[Mac04] L. Macaire : Exploitation de la couleur pour la segmentation et l'analyse d'images. *Habilitation à Diriger des Recherches*, 2004. 97

[Men01] G. Menegaz : DWT based non-parametric texture modeling. *In International Conference on Image Processing, ICIP*, volume 1, pages 173–176. IEEE, 2001. 14

[Mic95] A.A. Michelson : *Studies in optics*. Dover Pubns, 1995. 45

[MK97] GJ McLachlan et T. Krishnan : The EM algorithm and extensions, 1997. 31

[MS74] JL Mannos et D. Sakrison : The effects of a visual fidelity criterion of the encoding of images. *Information Theory, IEEE Transactions on*, 20(4):525–536, 1974. 46, 176

[MSS02] BS Manjunath, P. Salembier et T. Sikora : *Introduction to MPEG-7 : multimedia content description interface*, volume 1. John Wiley & Sons Inc, 2002. 67, 69

[NA03] A. Nealen et M. Alexa : Hybrid texture synthesis. *In Proceedings of the 14th Eurographics workshop on Rendering*, page 105. Eurographics Association, 2003. 27, 28, 176

[Nin09] Alexandre Ninassi : *De la perception locale des distorsions de codage à l'apprécіation globale de la qualité visuelle des images et des vidéos. Apport de l'attention visuelle dans le jugement de qualité*. Thèse de doctorat, Ecole polytechnique de l'Université de Nantes, 2009. 61

[NNBW09a] P. Ndjiki Nya, D. Bull et T. Wiegand : Perception-oriented video coding based on texture analysis and synthesis. *In International conference on Image Processing, ICIP*, pages 2273–2276, 2009. 67, 107

[NNBW09b] Patrick Ndjiki-Nya, Dave Bull et Thomas Wiegand : Perception-oriented video coding based on texture analysis and synthesis. *In ICIP*, pages 2273–2276, 2009. 77

[NNHSW05] Patrick Ndjiki-Nya, Tobias Hinz, Aljoscha Smolic et Thomas Wiegand : A generic and automatic content-based approach for improved h.264/mpeg4-avc video coding. *In International conference on Image Processing, ICIP*, pages 874–877, 2005. 70

[NNHW07] Patrick Ndjiki-Nya, Tobias Hinz et Thomas Wiegand : Generic and robust video coding with texture analysis and synthesis. *In ICME*, pages 1447–1450, 2007. 76, 77, 112, 113, 122, 155, 177

[NNKW04] P. Ndjiki-Nya, M. Kootz et T. Wiegand : Automatic detection of video synthesis related artifacts. *In Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 3, pages iii – 733–6 vol.3, mai 2004. 70, 122

[NNMB+03] Patrick Ndjiki-Nya, Bela Makai, Gabi Blattermann, Aljoscha Smolic, Heiko Schwarz et Thomas Wiegand : Improved h.264/avc coding using texture analysis and synthesis. *In International conference on Image Processing, ICIP*, pages 849–852, 2003. 69

[NNMS+03] P. Ndjiki-Nya, B. Makai, A. Smolic, H. Schwarz et T. Wiegand : Video coding using texture analysis and synthesis. *In Proceedings of Picture Coding Symposium, St. Malo, France*. Citeseer, 2003. 69, 70

[NNSW05] Patrick Ndjiki-Nya, Christoph Stüber et Thomas Wiegand : A new generic texture synthesis approach for enhanced h.264/mpeg4-avc video coding. *In VLBV*, pages 121–128, 2005. 74

[OBMC01] M. M Oliveira, B. Bowen, R. McKenna et Y. S Chang : Fast digital image inpainting. *In Proceedings of the International Conference on Visualization, Imaging and Image Processing (VIIP)*, pages 261–266, 2001. 38

[Oh09] B.T. Oh : *Texture processing for image/video coding and super-resolution applications*. Thèse de doctorat, UNIVERSITY OF SOUTHERN CALIFORNIA, 2009. 77

[OLL+03] E.P. Ong, W. Lin, Z. Lu, S. Yao, X. Yang et F. Moschetti : Low bit rate quality assessment based on perceptual characteristics. *In Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 3, pages III–189. IEEE, 2003. 64

[OLLY05] E. Ong, W. Lin, Z. Lu et S. Yao : Colour perceptual video quality metric. *In Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–1172. IEEE, 2005. 64

[OPH96] T. Ojala, M. Pietik
"ainen et D. Harwood : A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996. 82

[OR98] A. Ortega et K. Ramchandran : Rate-distortion methods for image and video compression. *Signal Processing Magazine, IEEE*, 15(6):23–50, 1998. 58

[OSS+08] B. T Oh, Y. Su, A. Segall, J. Kuo *et al.* : Synthesis-based texture coding for video compression with side information. *In 15th IEEE International Conference on Image Processing,ICIP*, pages 162–163, 2008. 75

[Per85] Ken Perlin : An image synthesizer. *In Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '85, pages 287–296, New York, NY, USA, 1985. ACM. 12

[PFH00] E. Praun, A. Finkelstein et H. Hoppe : Lapped textures. *In Proceedings of SIGGRAPH 2000*, pages 465–470. Citeseer, 2000. 25

[PGB03] P. Pérez, M. Gangnet et A. Blake : Poisson image editing. *In ACM Transactions on Graphics (TOG)*, volume 22, pages 313–318. ACM, 2003. 74, 77

[PGM+95] Rosalind Picard, Chris Graczyk, Steve Mann, Josh Wachman, Len Picard et Lee Campbell : Vison Texture, 1995. http ://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html. 40, 41, 91, 176, 178

[PL95] R. Paget et D. Longstaff : Texture Synthesis via a Non-parametric Markov Random Field. *Proceedings of DICTA–95, Digital Image Computing : Techniques and Applications, Anthony Maeder and Brian Lovell, Eds., Brisbane, Australia*, 1:547–552, 1995. 13

[PS00a] J. Portilla et E.P. Simoncelli : A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000. 68

[PS00b] Javier Portilla et Eero P. Simoncelli : A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comput. Vision*, 40(1):49–70, 2000. 14

[RMHN95] CA Rothwell, JL Mundy, W. Hoffman et V.D. Nguyen : Driving vision by topology. *In iscv*, page 395. Published by the IEEE Computer Society, 1995. 71

[RSB02] S.D. Rane, G. Sapiro et M. Bertalmio : Structure and texture filling-in of missing image blocks in wireless transmission and compression. *In Image*

Processing. 2002. Proceedings. 2002 International Conference on, volume 1, pages I–317. IEEE, 2002. 70

[SC94] J. Smith et S.F. Chang : Quad-tree segmentation for texture-based image query. *In Proceedings of the second ACM international conference on Multimedia*, pages 279–286. ACM, 1994. 83

[SDW01] S. Soatto, G. Doretto et Y.N. Wu : Dynamic textures. *In Proceedings. Eighth IEEE International Conference on Computer Vision, ICCV*, volume 2, pages 439–446. IEEE, 2001. 13

[SHB93] M. Sonka, V. Hlavac et R. Boyle : Image Processing, Analysis, and Machine Vision. *In Chapman & Hall*, 1993. 99

[SJ88] H.G. Schuster et W. Just : *Deterministic chaos*. Wiley Online Library, 1988. 25

[SLG+08] F. Smach, C. Lemaître, J.P. Gauthier, J. Miteran et M. Atri : Generalized Fourier descriptors with applications to objects recognition in SVM context. *Journal of Mathematical Imaging and Vision*, 30(1):43–71, 2008. 84

[Sma09] Féthi Smach : *Étude et implémentation des Descripteurs de Fourier Généralisés combinés aux SVM en vue de la reconnaissance d'objets en temps réel*. Thèse de doctorat, University of Bristol, 2009. 84, 85

[SMW07] H. Schwarz, D. Marpe et T. Wiegand : Overview of the scalable video coding extension of the h. 264/avc standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(9):1103–1120, 2007. 141

[SP95] M. Szummer et R.W. Picard : Temporal texture modeling. *In International Conference on Image Processing, ICIP*, volume 3, pages 823–826. IEEE, 1995. 13

[SPD07] Muath Sabha, Pieter Peers et Philip Dutré : Texture synthesis using exact neighborhood matching. *Computer Graphics Forum*, 26(2):131–142, 2007. 23, 24, 40, 176

[SS00] H. Y Shum et R. Szeliski : Systems and experiment paper : Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130, 2000. 30

[SSW88] P.K. Sahoo, S. Soltani et AKC Wong : A survey of thresholding techniques. *Computer vision, graphics, and image processing*, 41(2):233–260, 1988. 96

[SW98] G.J. Sullivan et T. Wiegand : Rate-distortion optimization for video compression. *Signal Processing Magazine, IEEE*, 15(6):74–90, 1998. 58

[SW02] H. Schwarz et T. Wiegand : The emerging JVT/H. 26L video coding standard. *Proc. of IBC 2002*, 2002. 53

[TBS06] T.K. Tan, C.S. Boon et Y. Suzuki : Intra prediction by template matching. *In Image Processing, 2006 IEEE International Conference on*, pages 1693–1696. IEEE, 2006. 65

[TMT00] M. Topi, P. Matti et O. Timo : Texture classification by multi-predicate local binary pattern operators. *In Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 3, pages 939–942. IEEE, 2000. 82

[TZL+02] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo et H. Y Shum : Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics*, 21(3):665–672, 2002. 2, 21, 22, 23, 34, 112, 126

[VO03] L.A. Vese et S.J. Osher : Modeling textures with total variation minimization and oscillating patterns in image processing. *Journal of Scientific Computing*, 19(1):553–572, 2003. 39

[WA94] J.Y.A. Wang et E.H. Adelson : Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994. 67

[WBSS04] Z. Wang, A. C. Bovik, H. R. Sheikh et E. P. Simoncelli : Image quality assessment : From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 62

[WHZ+08] L.Y. Wei, J. Han, K. Zhou, H. Bao, B. Guo et H.Y. Shum : Inverse texture synthesis. *In ACM SIGGRAPH 2008 papers*, pages 1–9. ACM, 2008. 35

[WL00] Li-Yi Wei et Marc Levoy : Fast texture synthesis using tree-structured vector quantization. *In SIGGRAPH '00 : Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 2, 16, 18, 19, 20, 21, 22, 23, 25, 28, 33, 34, 40, 41, 66, 94, 112, 114, 126, 132, 175, 176, 178

[WL02] L.Y. Wei et M. Levoy : Order-independent texture synthesis. *Computer Science Department, Stanford University*, 2002. 22, 23

[WLB04] Z. Wang, L. Lu et A.C. Bovik : Video quality assessment based on structural distortion measurement. *Signal processing : Image communication*, 19(2):121–132, 2004. 63

[Wor96] S. Worley : A cellular texture basis function. *In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 291–294. ACM, 1996. 12

[WSB02] Zhou Wang, Hamid R. Sheikh et Alan C. Bovik : No-reference perceptual quality assessment of jpeg compressed images. *In Proceedings of IEEE 2002 International Conferencing on Image Processing*, pages 477–480, 2002. 62

[WSB03] Z. Wang, E.P. Simoncelli et A.C. Bovik : Multiscale structural similarity for image quality assessment. *In Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1398–1402. IEEE, 2003. 63

[WSJ+03] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini et G.J. Sullivan : Rate-constrained coder control and comparison of video coding standards. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):688–703, 2003. 58, 66

[WSWX06] C. Wang, X. Sun, F. Wu et H. Xiong : Image compression with structure-aware inpainting. *In 2006 IEEE International Symposium on Circuits and Systems, ISCAS*, page 4, 2006. 70

[WZ02] Y. Wang et S.C. Zhu : A generative method for textured motion, Analysis and synthesis. *Computer Vision ECCV 2002*, pages 583–598, 2002. 13

[XSW10] Zhiwei Xiong, Xiaoyan Sun et Feng Wu : Block-based image compression with parameter-assistant inpainting. *IEEE Transactions on Image Processing*, 19(6):1651–1657, 2010. 71, 117

[YHS] H. Yamauchi, J. Haber et H. P Seidel : Image restoration using multiresolution texture synthesis and image inpainting. *transfer*, 8:12. 40

[YHS03] H. Yamauchi, J. Haber et H.P. Seidel : Image restoration using multiresolution texture synthesis and image inpainting. *In Computer Graphics International, 2003. Proceedings*, pages 120–125. IEEE, 2003. 7

[ZSWL07] C. Zhu, X. Sun, F. Wu et H. Li : Video coding with spatio-temporal texture synthesis. *In 2007 IEEE International Conference on Multimedia and Expo*, pages 112–115, 2007. 68, 70, 71, 73, 74, 75, 76, 77, 113, 118, 177

[ZSWL08] C. Zhu, X. Sun, F. Wu et H. Li : Video coding with spatio-temporal texture synthesis and edge-based inpainting. *In IEEE International Conference on Multimedia and Expo*, pages 813–816. IEEE, 2008. 113, 155

[ZWM98] S.C. Zhu, Y. Wu et D. Mumford : Filters, random fields and maximum entropy (FRAME) : Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998. 12

INSA de Rennes
Service des formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse : Mise en oeuvre de techniques d'analyse / Synthèse de texture dans un schéma de compression vide

Nom Prénom de l'auteur : RACAPE Fabien

Membres du jury :
- Monsieur NDJIKI-NYA
- Monsieur SALEMBIER
- Monsieur DEFORGES
- Madame BABEL
- Monsieur BASKURT
- Madame MORIN
- Monsieur THOREAU
- Monsieur VIERON

Président du jury : A. BASKURT

Date de la soutenance : 14/11/2011

Reproduction de la thèse soutenue :

- [x] Thèse pouvant être reproduite en l'état
- [ ] Thèse ne pouvant être reproduite
- [ ] Thèse pouvant être reproduite après corrections suggérées

Le Directeur,

M'Hamed DRISSI

Rennes, le 14/11/2011

Signature du Président du jury

Résumé

Cette thèse s'inscrit dans le contexte des schémas de compression vidéo de nouvelles générations. Elle vise plus particulièrement à coder plus efficacement les régions texturées des images et séquences vidéo que les schémas actuels. Ces zones sont souvent dégradées lors de codage à bas débit, provoquant des aplats visuellement dérangeants. Ce travail est fondé sur les propriétés du système visuel humain, qui préfèrera une zone texturée synthétisée avec des détails, même un peu éloignée de la réalité, plutôt que des aplats. L'idée est ici d'adapter les algorithmes de synthèse de texture de la littérature, afin de reconstruire, au décodeur, des régions qui n'auront pas été intégralement transmises.

L'approche est construite de manière à être utilisée conjointement avec les standards de compression actuels ou futurs. L'analyse de la séquence source, côté encodeur, utilise des outils de segmentation et de caractérisation de texture, afin de localiser les régions candidates pour la synthèse. Les régions qui ne sont pas synthétisables sont encodées classiquement par le codeur joint, elles seront décodées et serviront potentiellement d'échantillons de départ pour la synthèse des zones manquantes.

L'ensemble des outils ont été développés et adaptés dans l'optique principale de proposer une chaîne cohérente. L'analyse des textures comportant des outils de segmentation et de caractérisation permettant de paramétrer les algorithmes de synthèse. Aussi la solution proposée inclut l'utilisation de deux types de synthèse : une version orientée « pixel » et l'autre orientée « patch ». Une première approche est présentée pour un codage intra image. Le schéma est ensuite couplé à une méthode d'estimation et de modélisation affine de mouvement par région, afin d'optimiser le traitement des textures rigides et de synthétiser les régions déformables.

Fondé sur des outils de synthèse, le schéma est difficilement estimable à l'aide de critères objectifs. A qualité visuelle comparable, il permet, par exemple, de préserver jusqu'à 33% de débit, comparé à l'encodage de H.264/AVC, sur différentes séquences SD et CIF.

Abstract

This thesis comes within the scope of new generation video compression schemes. In particular, it aims at improving the coding efficiency for textured regions in images and videos. At low bit-rate, textures are degraded, resulting in visually annoying flat tints. The basic assumption is based on the human visual system properties, which prefer synthesized details to flat color, even if the output surface is not exactly the source texture. In this work, texture synthesis algorithms from the literature are adapted in order to fit the coding context: filling textures which are not entirely transmitted.

This approach is designed to be jointly used with current and future standard compression schemes. At encoder side, texture analysis includes segmentation and characterization tools, in order to localize candidate regions for synthesis. The corresponding areas are not encoded. The decoder fills them using texture synthesis. The remaining regions in images are classically encoded. They can potentially serve as input for texture synthesis.

The chosen tools are developed and adapted with an eye to ensuring the coherency of the whole scheme. Thus, a texture characterization step provides required parameters to the texture synthesizer. Two texture synthesizers, including a pixel-based and a patch-based approach, are used on different types of texture, complementing each other. A first scheme is proposed for intra frame coding. Then, a temporal method is developed. The scheme is coupled with a motion estimator in order to segment coherent regions and to interpolate rigid motions using an affine model. Inter frame adapted synthesis is therefore used for non-rigid texture regions.

Assessing the quality of decoded frames by such schemes, using objective methods, is problematic. Results on bit-rate savings are presented with the assumption of similar visual quality. Thus, now subjective tests provide for now the assessment. At comparable visual quality, up to 33% bit-rate is preserved, compared to H.264/AVC, on many SD and CIF sequences.

N° d'ordre : D11 - 26

Institut National des Sciences Appliquées de Rennes
20, Avenue des Buttes de Coësmes • CS 70839 • F-35708 Rennes Cedex 7
Tel : 02 23 23 82 00 - Fax : 02 23 23 83 96